Санкт-Петербургский Государственный университет
аэрокосмического приборостроения

Основы программирования

Направление 09.03.03 Прикладная информатика

Степанов
Александр Георгиевич
georgich_spb@mail.ru

2017

Введение. Литература по курсу.

1. Степанов А.Г., Мичурин С.В. Информатика и программирование. Учебное пособие/СПбГУАП. СПб., 2004. – 120 с.
2. Информатика. Применение программ пакета Microsoft Office. Учебно-методическое пособие. /Н.В. Зуева, Н.С. Медведева, О.И. Москалева, А.Г. Степанов. ГУАП, 2007.
3. Информатика. Программирование на языке VBA. Зуева Н.В. и др. Методические указания к выполнению лабораторных работ. ГУАП, 2007 г.
4. Информатика. Методические указания по выполнению курсовой работы. ГУАП, 2007.

Дополнительная литература по курсу

1. Уокенбах Д. Подробное руководство по созданию формул в Excel 2002. : Пер. с англ. — М. : Издательский дом "Вильяме", 2002. – 624 с.
2. Гарнаев А. Ю. Самоучитель VBA. – СПб.: БХВ - Санкт-Петербург, 1999. – 512 с.
3. Малышев С.А. Самоучитель VBA. Как это делается в Word, Excel, Access. – СПб: Наука и техника, 2001. – 496 стр.
4. Visual Basic 6.0: Пер. с англ. – СПб.: БХВ-Петербург, 2002. – 992 стр.
5. Браун С. Visual Basic 5 с самого начала. – СПб: Питер, 1998. – 320 с.
6. Буч Г. Объектно-ориентированный анализ и проектирование с примерами приложений на C++, 2-е изд. : Пер. с англ. – М.: «Издательство Бином», СПб.: «Невский диалект», 2000. – 560 с.

1. Язык программирования VBA

В разделе рассматривается:

- Элементарное взаимодействие Excel и VBA
- Отладка и выполнение программы в среде VBA
- Обмен данными между Excel и VBA

1.1. Элементарное взаимодействие Excel и VBA

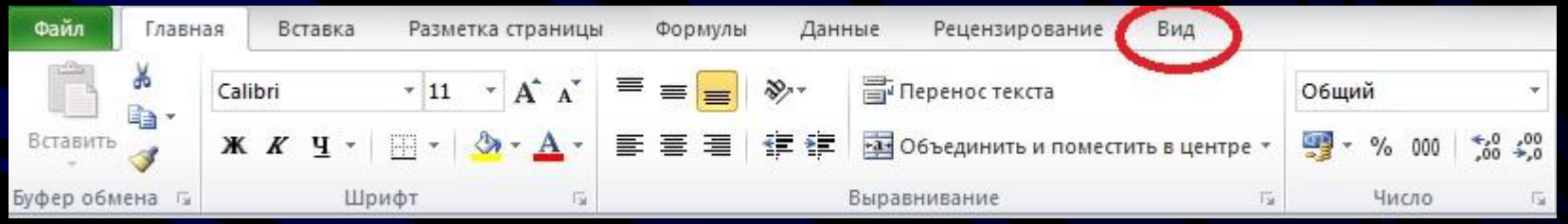
В подразделе рассматривается:

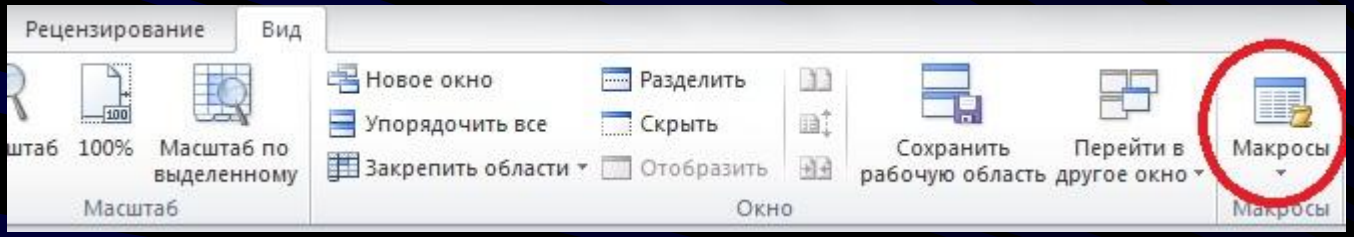
- Создание и выполнение макросов Excel
- Разработка пользовательской таблицы средствами процессора Excel
- Интегрированная среда разработки VBA
- Типы записи ссылок в Excel
- Анализ текста созданного макроса

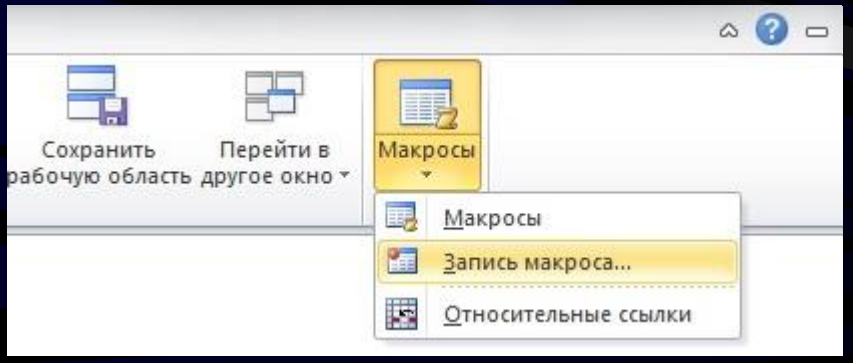
1.1.1. Создание и выполнение макросов Excel

- *Макросом* обычно называют файл, хранящий последовательность действий, заданных пользователем
- Каждый макрос должен иметь собственное *имя*.
- По своей сути макрос представляет собой *программу* и может быть создан автоматически в специальном режиме работы программной системы (в том числе и *Excel*) или как результат программирования в терминах языка системы

Создание и выполнение макросов Excel

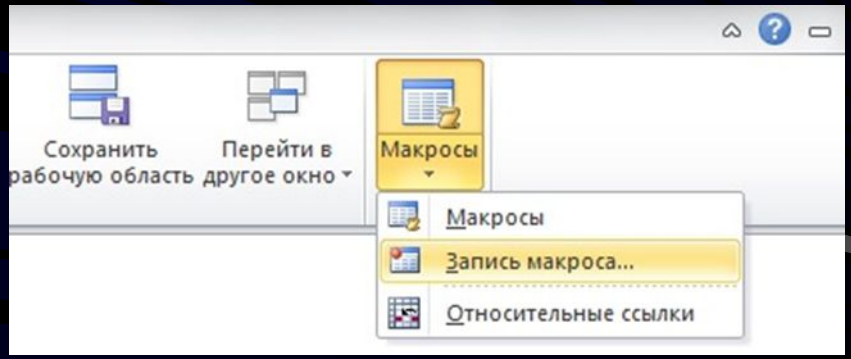






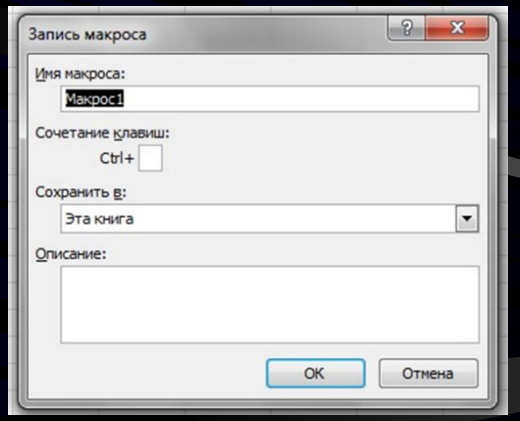
Создание и выполнение макросов Excel

- Для создания макроса в Excel легче всего использовать автоматический режим его создания, вызываемый из главного меню системы командами **СЕРВИС, Макрос**.



Создание и выполнение макросов Excel

- Если в меню **СЕРВИС, Макрос** выбрать пункт **Начать запись…**, то откроется диалоговое окно, позволяющее задать имя макроса и, при желании, комбинацию клавиш, с помощью которой он также может вызван в обход пункта меню **Макросы…**.

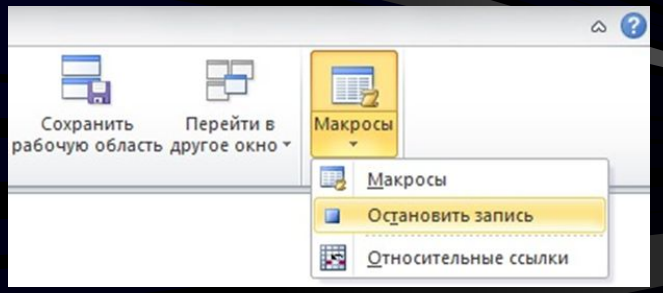


Создание и выполнение макросов Excel

- По умолчанию система предлагает стандартное имя **Макрос#**.
- Во избежание недоразумений старайтесь задавать собственные имена макросов, отличные от стандартных.

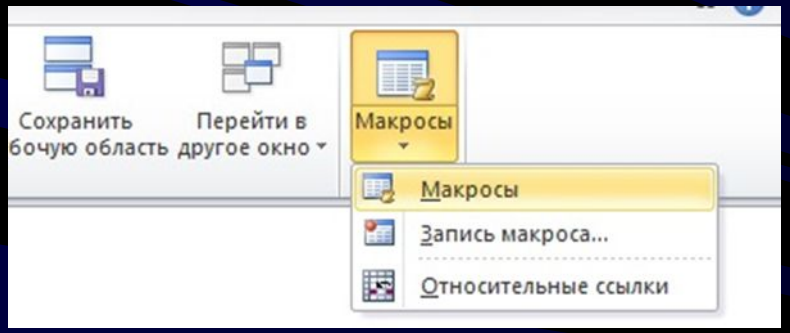
Создание и выполнение макросов Excel

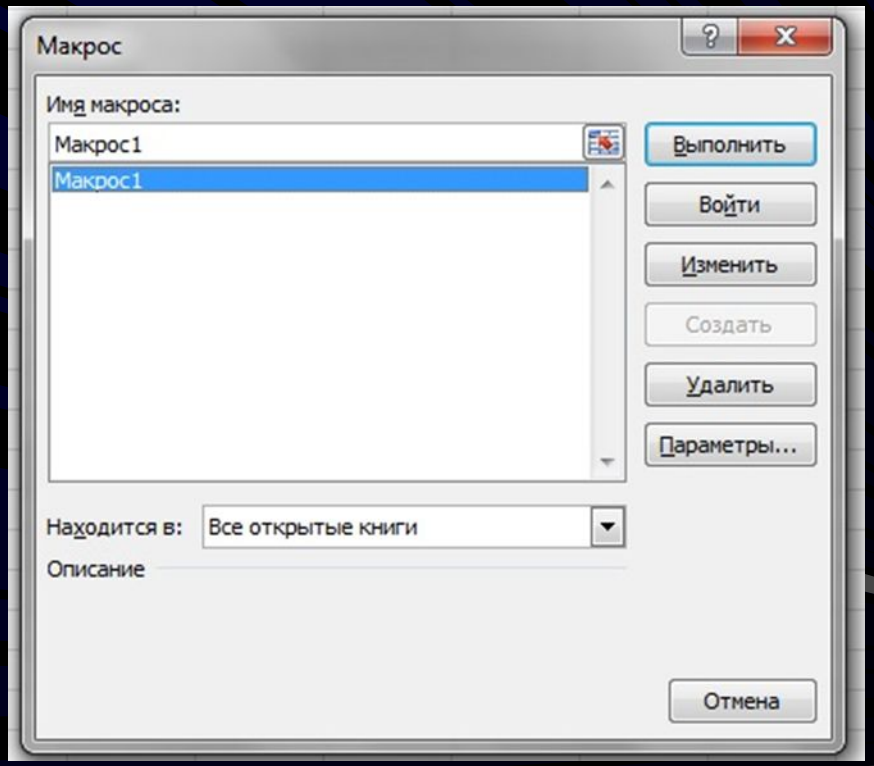
- Начиная с этого момента все действия с рабочей книгой дополнительно записываются в файл макроса
- Остановить запись макроса можно кнопкой **Остановить запись** дополнительно открывшейся панели инструментов или через аналогичный пункт главное меню **СЕРВИС**, **Макрос**.



Создание и выполнение макросов Excel

- К записанному макросу можно обратиться через главное меню





1.1.2. Разработка пользовательской таблицы средствами процессора Excel

Мы предполагаем, что вы:

- в состоянии придумать собственную *пользовательскую таблицу*, данные в которой организованы по строкам и столбцам, имеют вполне определенный практический смысл и требуют некой обработки, в частности, вычислений
- знакомы со способами ее оформления (шрифт, фон, рамки)

Разработка пользовательской таблицы средствами процессора Excel

	A	B	C	D	E
1	*Фамилия, И.О.*	*Начислено*	*Налог*	*К выдаче*	
2	Иванов В.Н.	1 234,00р.			
3	Трофимова Л.А.	1 234,00р.			
4	Семенова Е.Г.	1 000,00р.			
5	Степанов А.Г.	900,00р.			
6	**Итого**	4 368,00р.			
7	Ставка подоходного налога		12,00%		
8					

Исходные данные

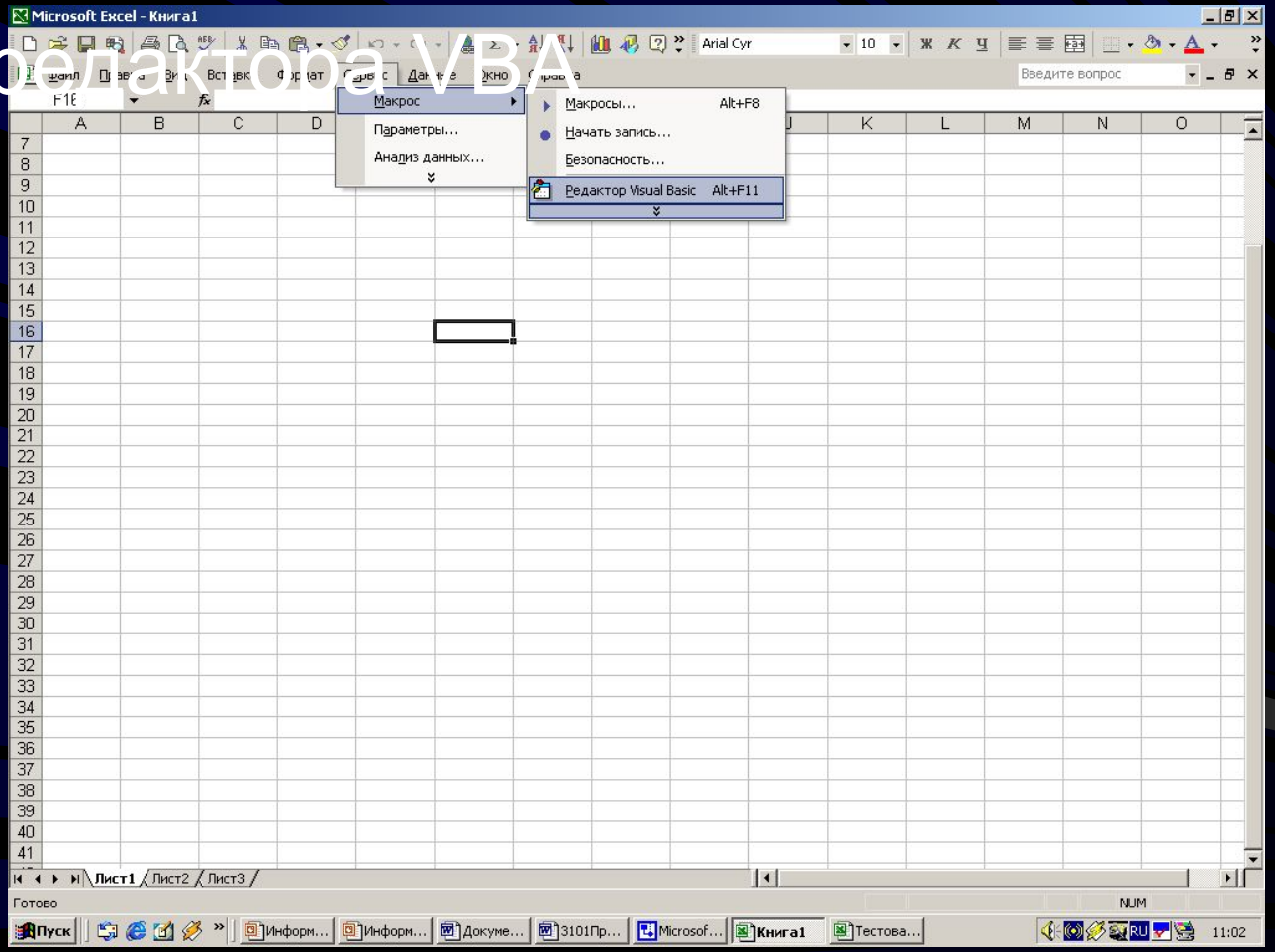
Разработка пользовательской таблицы средствами процессора Excel

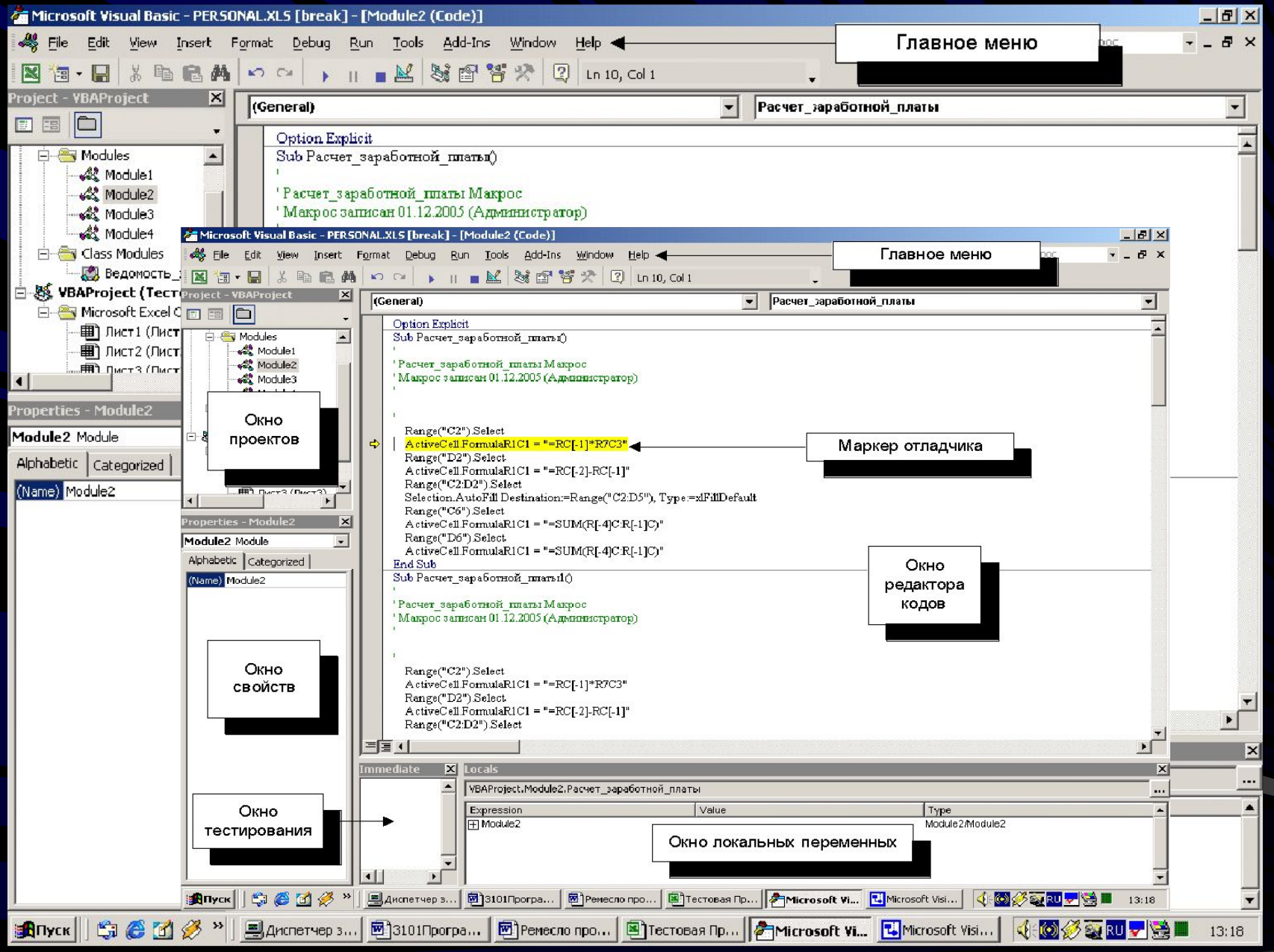
	A	B	C	D
1	*Фамилия, И.О.*	*Начислено*	*Налог*	*К выдаче*
2	Иванов В.Н.	1 234,00р.	148,08р.	1 085,92р.
3	Трофимова Л.А.	1 234,00р.	148,08р.	1 085,92р.
4	Семенова Е.Г.	1 000,00р.	120,00р.	880,00р.
5	Степанов А.Г.	900,00р.	108,00р.	792,00р.
6	**Итого**	4 368,00р.	524,16р.	3 843,84р.
7	Ставка подоходного налога		12,00%	

Результаты программирования в Excel

1.1.3. Интегрированная среда разработки VBA

- Запуск редактора VBA




Главное меню
Маркер отладчика
Окно редактора кодов
Окно проектов
Окно свойств
Окно тестирования
Окно локальных переменных

1.1.4. Типы записи ссылок в Excel

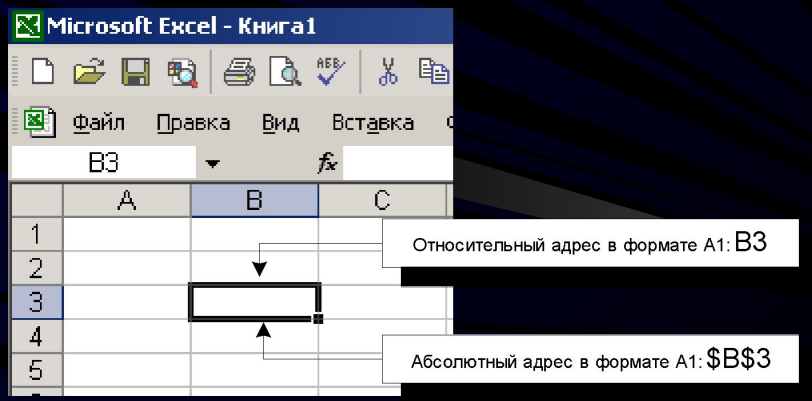
- Необходимо принять во внимание существование двух возможных типов записи ссылок на ячейки в Excel: A1 и R1C1.
- По умолчанию при программировании формул используется стиль A1, для которого адрес каждой ячейки представляет собой строку символов, содержащую имя столбца и номер строки.

Типы записи ссылок в Excel

- При записи макросов Excel использует тип ссылки R1C1.
- В обозначении типа присутствуют первые буквы английских слов Row (строка) и Column (колонка).

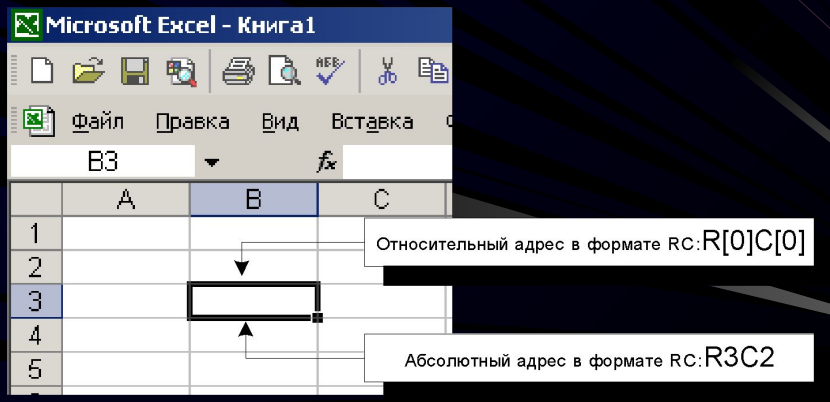
Типы записи ссылок в Excel

- Использование стиля А1 позволяют организовать относительную и абсолютную адресацию к ячейкам таблицы (за счет введения в строку символа $).



Типы записи ссылок в Excel

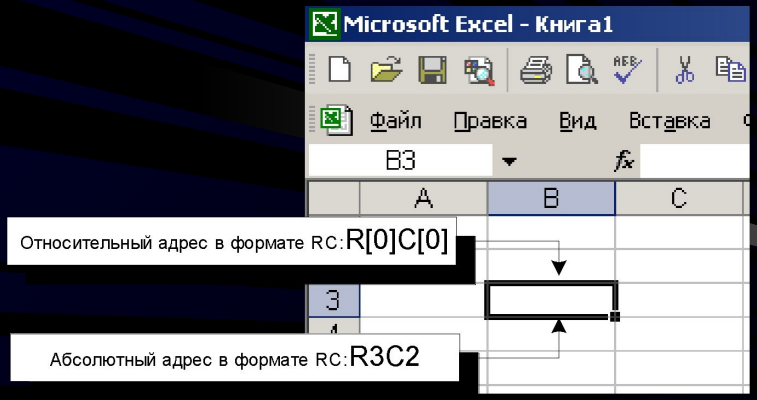
- При использовании абсолютной адресации после символов R и C указывается собственно номер строки и столбца. Так, например, ячейка B3 имеет адрес R3C2.



Обратите внимание на то, что, в отличие от типа A1, при использовании типа ссылок R1C1 сначала записывается строка, а потом столбец.

Типы записи ссылок в Excel

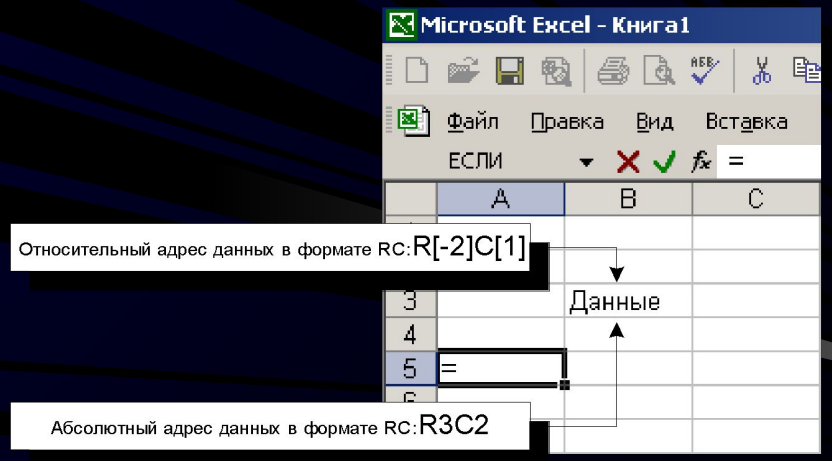
- При использовании относительной адресации в стиле R1C1 после обозначения строки или колонки в квадратных скобках указывается смещение по отношению к текущей ячейке.



Типы записи ссылок в Excel

- Так, например, если данные находятся в ячейке B3, а ссылка на нее программируется в ячейке A5, то в формуле она запишется как R[-2]C[1]

Эта запись может интерпретироваться как обращение к ячейке, находящейся на две строки выше и одну колонку правее текущей.

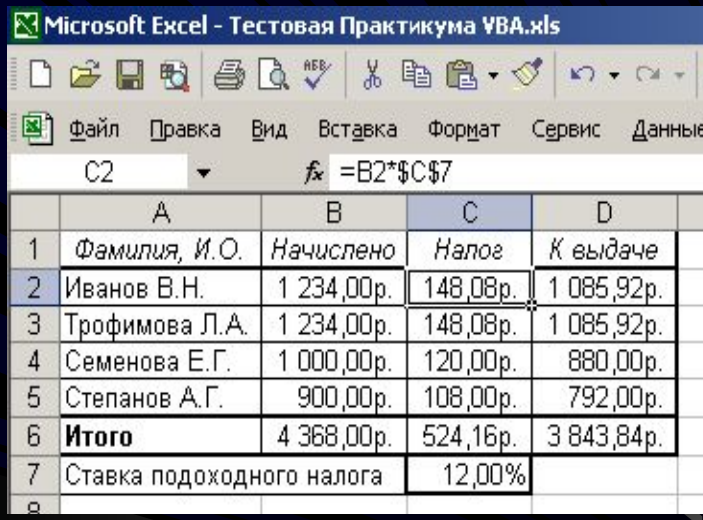


Типы записи ссылок в Excel

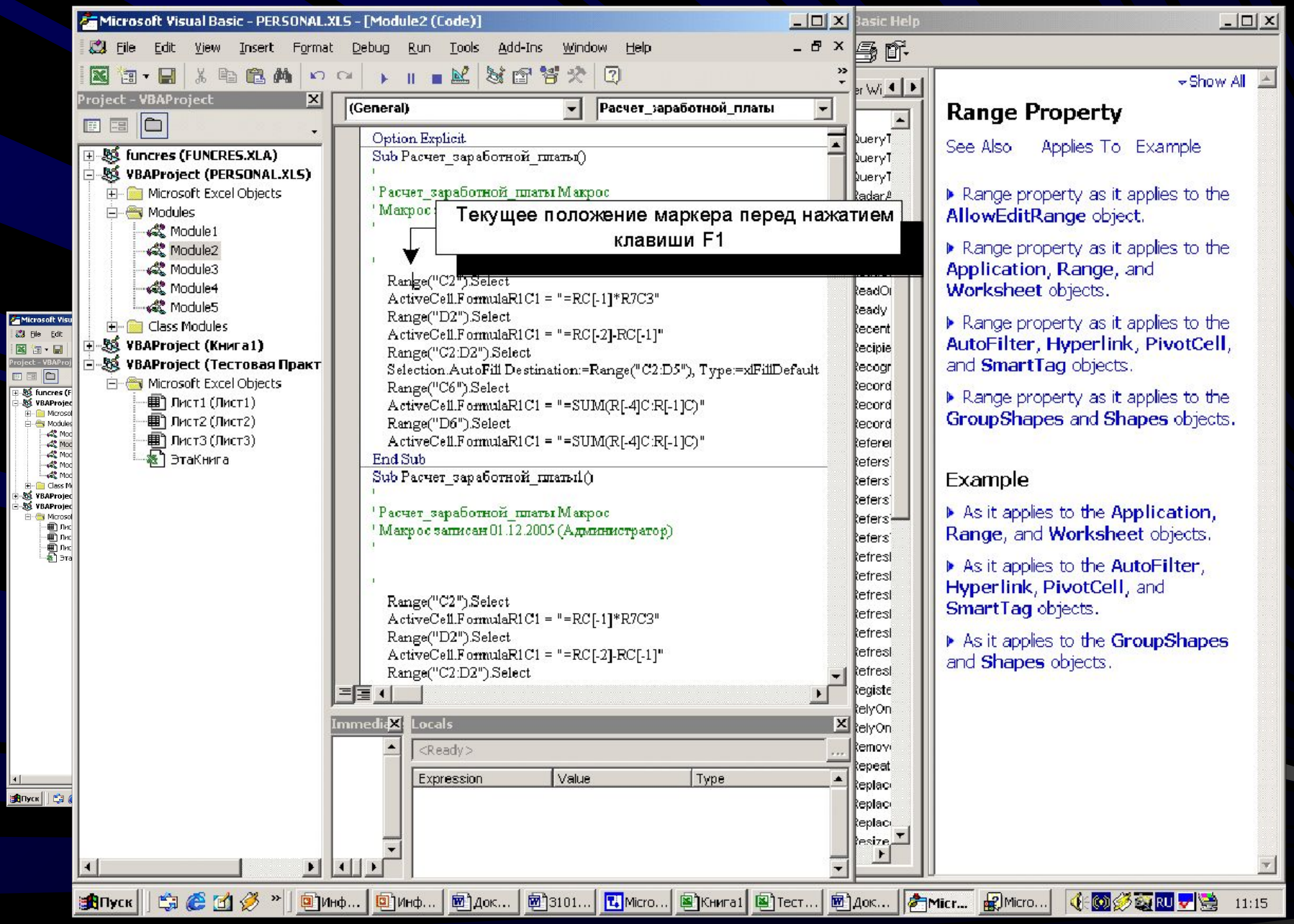
- Соответственно запись R[2]C[-1] означает обращение к ячейке на две строки ниже и одну колонку левее (по отношению к активной ячейке A5 такая ячейка не существует).

1.1.5. Анализ текста созданного макроса

```
Sub Расчет_заработной_платы()
'
' Расчет _заработной_платы Макрос
' Макрос записан 01.12.2005 (Администратор)
'
'
    Range("C2").Select
    ActiveCell.FormulaR1C1 = "=RC[-1]*R7C3"
    Range("D2").Select
    ActiveCell.FormulaR1C1 = "=RC[-2]-RC[-1]"
    Range("C2:D2").Select
    Selection.AutoFill Destination:=Range("C2:D5"), Type:=xlFillDefault
    Range("C6").Select
    ActiveCell.FormulaR1C1 = "=SUM(R[-4]C:R[-1]C)"
    Range("D6").Select
    ActiveCell.FormulaR1C1 = "=SUM(R[-4]C:R[-1]C)"
End Sub
```



	A	B	C	D
1	*Фамилия, И.О.*	*Начислено*	*Налог*	*К выдаче*
2	Иванов В.Н.	1 234,00р.	148,08р.	1 085,92р.
3	Трофимова Л.А.	1 234,00р.	148,08р.	1 085,92р.
4	Семенова Е.Г.	1 000,00р.	120,00р.	880,00р.
5	Степанов А.Г.	900,00р.	108,00р.	792,00р.
6	**Итого**	4 368,00р.	524,16р.	3 843,84р.
7	Ставка подоходного налога		12,00%	

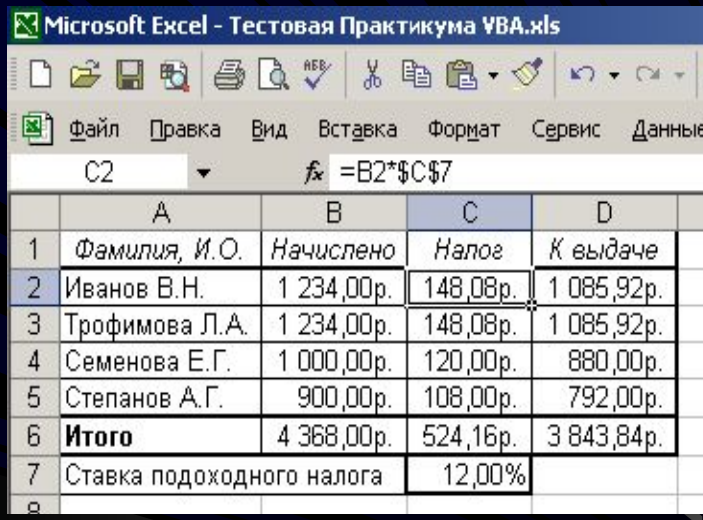

Microsoft Visual Basic - PERSONAL.XLS - [Module2 (Code)]
File Edit View Insert Format Debug Run Tools Add-Ins Window Help
Project - VBAProject
funcres (FUNCRES.XLA)
VBAProject (PERSONAL.XLS)
Microsoft Excel Objects
Modules
Module1
Module2
Module3
Module4
Module5
Class Modules
VBAProject (Книга1)
VBAProject (Тестовая Практ
Microsoft Excel Objects
Лист1 (Лист1)
Лист2 (Лист2)
Лист3 (Лист3)
ЭтаКнига
(General)
Расчет_заработной_платы
Option Explicit
Sub Расчет_заработной_платы()
' Расчет_заработной_платы Макрос
' Макрос
Текущее положение маркера перед нажатием клавиши F1
Range("C2").Select
ActiveCell.FormulaR1C1 = "=RC[-1]*R7C3"
Range("D2").Select
ActiveCell.FormulaR1C1 = "=RC[-2]-RC[-1]"
Range("C2:D2").Select
Selection.AutoFill Destination:=Range("C2:D5"), Type:=xlFillDefault
Range("C6").Select
ActiveCell.FormulaR1C1 = "=SUM(R[-4]C:R[-1]C)"
Range("D6").Select
ActiveCell.FormulaR1C1 = "=SUM(R[-4]C:R[-1]C)"
End Sub
Sub Расчет_заработной_платы1()
' Расчет_заработной_платы Макрос
' Макрос записан 01.12.2005 (Администратор)
Range("C2").Select
ActiveCell.FormulaR1C1 = "=RC[-1]*R7C3"
Range("D2").Select
ActiveCell.FormulaR1C1 = "=RC[-2]-RC[-1]"
Range("C2:D2").Select
Immedia
Locals
<Ready>
Expression
Value
Type
Basic Help
Show All
Range Property
See Also Applies To Example
Range property as it applies to the AllowEditRange object.
Range property as it applies to the Application, Range, and Worksheet objects.
Range property as it applies to the AutoFilter, Hyperlink, PivotCell, and SmartTag objects.
Range property as it applies to the GroupShapes and Shapes objects.
Example
As it applies to the Application, Range, and Worksheet objects.
As it applies to the AutoFilter, Hyperlink, PivotCell, and SmartTag objects.
As it applies to the GroupShapes and Shapes objects.
Пуск
11:15

Анализ текста созданного макроса

- Range (диапазон). Возникает при выделении
- ActiveCell (активная ячейка). Возвращает объект Range
- FormulaR1C1. Свойство, возвращающее или задающее формулу типа R1C1 в активную ячейку
- Selection (выделение). Свойство, возвращающее выделенный объект
- AutoFill (автозаполнение). Метод, осуществляющий заполнение выделенных ячеек

1.1.5. Анализ текста созданного макроса

```
Sub Расчет_заработной_платы()
'
' Расчет _заработной_платы Макрос
' Макрос записан 01.12.2005 (Администратор)
'
'
    Range("C2").Select
    ActiveCell.FormulaR1C1 = "=RC[-1]*R7C3"
    Range("D2").Select
    ActiveCell.FormulaR1C1 = "=RC[-2]-RC[-1]"
    Range("C2:D2").Select
    Selection.AutoFill Destination:=Range("C2:D5"), Type:=xlFillDefault
    Range("C6").Select
    ActiveCell.FormulaR1C1 = "=SUM(R[-4]C:R[-1]C)"
    Range("D6").Select
    ActiveCell.FormulaR1C1 = "=SUM(R[-4]C:R[-1]C)"
End Sub
```

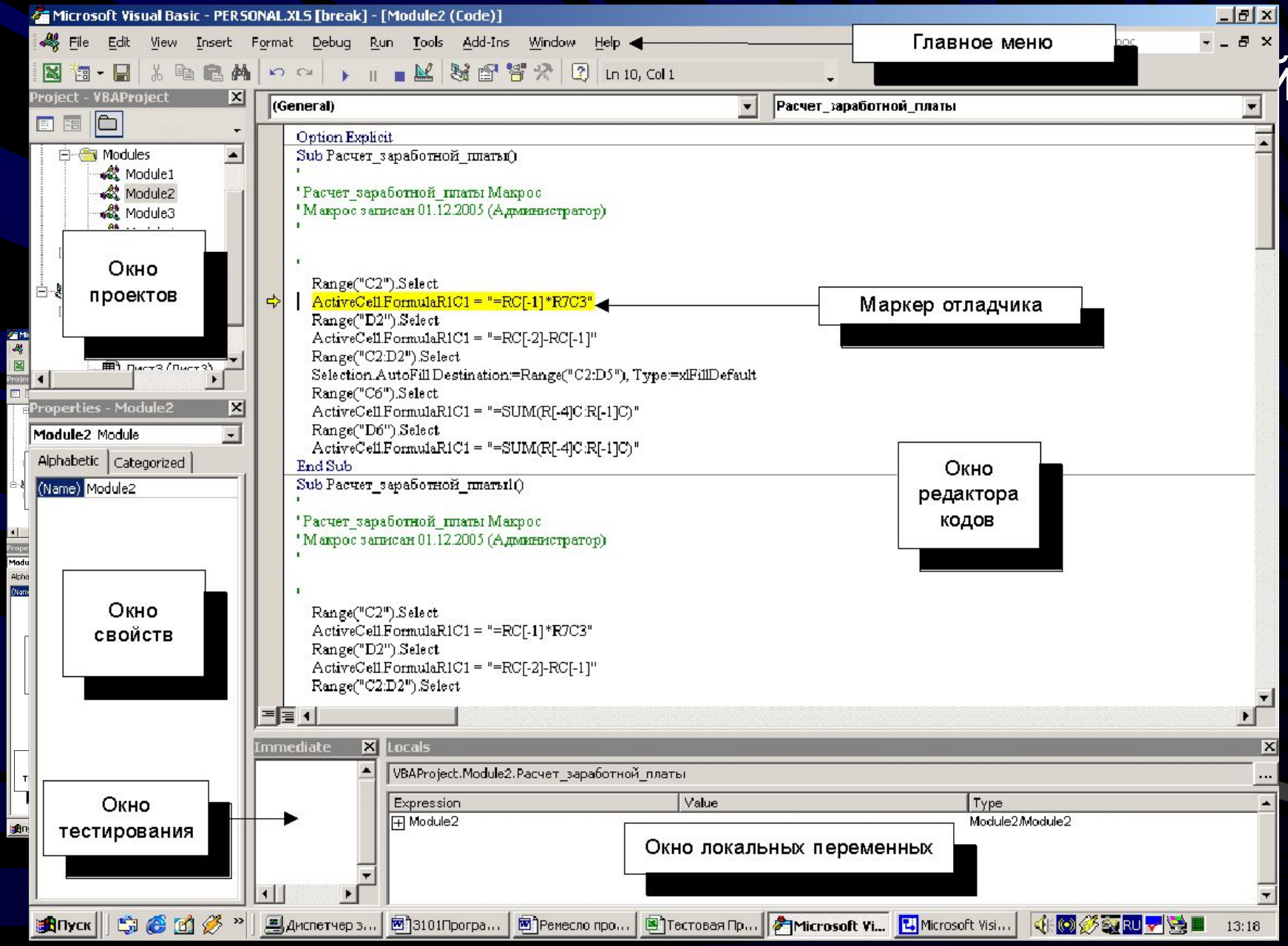


	A	B	C	D
1	*Фамилия, И.О.*	*Начислено*	*Налог*	*К выдаче*
2	Иванов В.Н.	1 234,00р.	148,08р.	1 085,92р.
3	Трофимова Л.А.	1 234,00р.	148,08р.	1 085,92р.
4	Семенова Е.Г.	1 000,00р.	120,00р.	880,00р.
5	Степанов А.Г.	900,00р.	108,00р.	792,00р.
6	**Итого**	4 368,00р.	524,16р.	3 843,84р.
7	Ставка подоходного налога		12,00%	

1.2. Отладка и выполнение программы в среде VBA

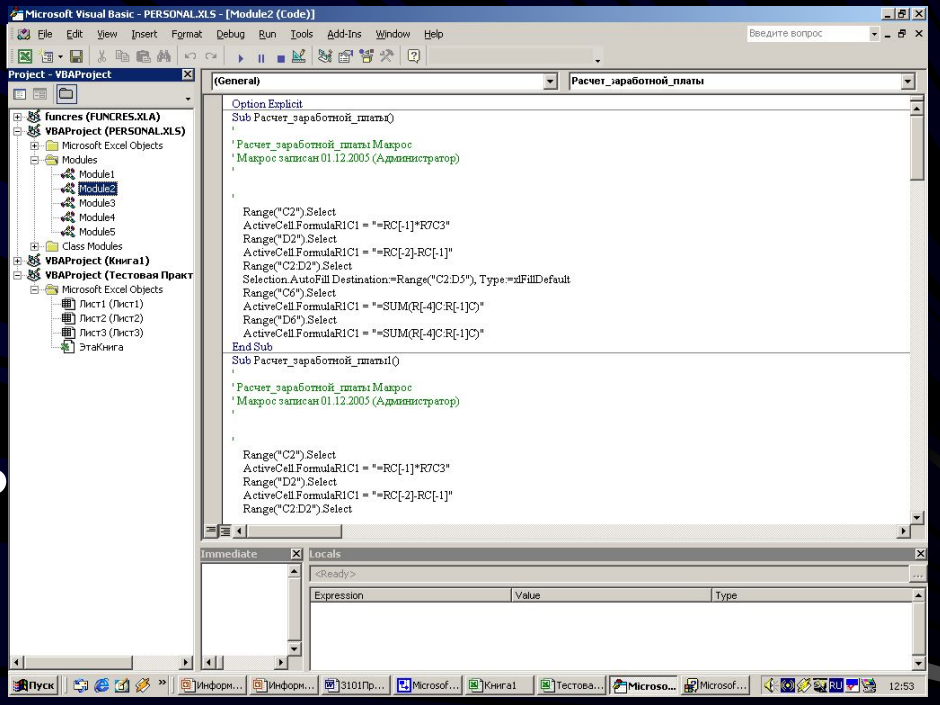
В подразделе рассматривается:

- Назначение окон интегрированной среды разработки VBA
- Выполнение программы в автоматическом режиме
- Выполнение программы в режиме отладки


Microsoft Visual Basic - PERSONAL.XLS [break] - [Module2 (Code)]
File Edit View Insert Format Debug Run Tools Add-Ins Window Help
Главное меню
Ln 10, Col 1
Project - VBAProject
Modules
Module1
Module2
Module3
Окно проектов
Properties - Module2
Module2 Module
Alphabetic Categorized
(Name) Module2
Окно свойств
(General)
Расчет_заработной_платы
Option Explicit
Sub Расчет_заработной_платы()
'Расчет_заработной_платы Макрос
'Макрос записан 01.12.2005 (Администратор)
Range("C2").Select
ActiveCell.FormulaR1C1 = "=RC[-1]*R7C3"
Range("D2").Select
ActiveCell.FormulaR1C1 = "=RC[-2]-RC[-1]"
Range("C2:D2").Select
Selection.AutoFill Destination:=Range("C2:D5"), Type:=xlFillDefault
Range("C6").Select
ActiveCell.FormulaR1C1 = "=SUM(R[-4]C:R[-1]C)"
Range("D6").Select
ActiveCell.FormulaR1C1 = "=SUM(R[-4]C:R[-1]C)"
End Sub
Sub Расчет_заработной_платы1()
'Расчет_заработной_платы Макрос
'Макрос записан 01.12.2005 (Администратор)
Range("C2").Select
ActiveCell.FormulaR1C1 = "=RC[-1]*R7C3"
Range("D2").Select
ActiveCell.FormulaR1C1 = "=RC[-2]-RC[-1]"
Range("C2:D2").Select
Маркер отладчика
Окно редактора кодов
Immediate
Locals
VBAProject.Module2.Расчет_заработной_платы
Expression
Value
Type
Module2
Module2/Module2
Окно локальных переменных
Окно тестирования
Пуск
Диспетчер з...
3101Програ...
Ремесло про...
Тестовая Пр...
Microsoft Vi...
Microsoft Visi...
13:18

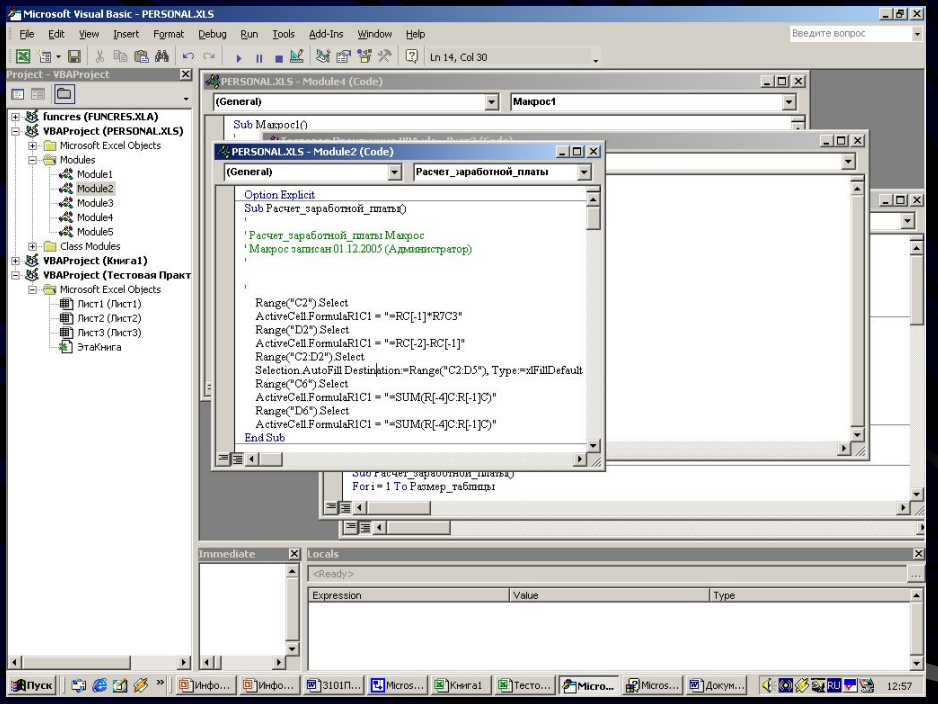
Назначение окон интегрированной среды разработки VBA

- Окно *проектов* содержит список фор модулей текущего проекта.
- *Проект* – набор файлов, используемь для построения приложений.



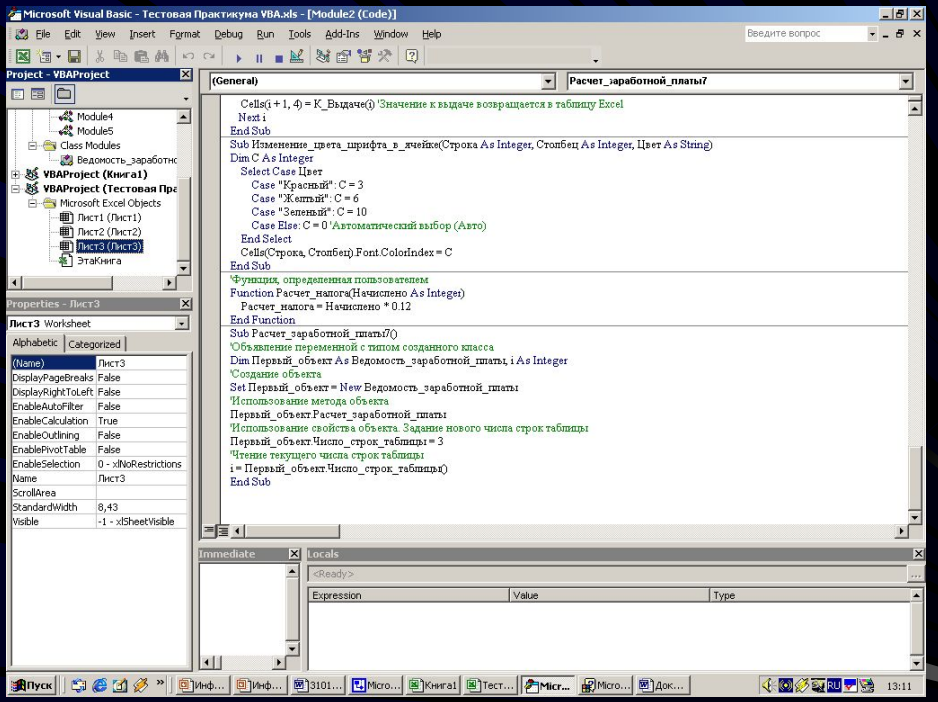
Назначение окон интегрированной среды разработки VBA

- Окно *редактора кодов* служит для редактирования программного кода приложения. Для каждой формы и каждого модуля кода создается свое окно.



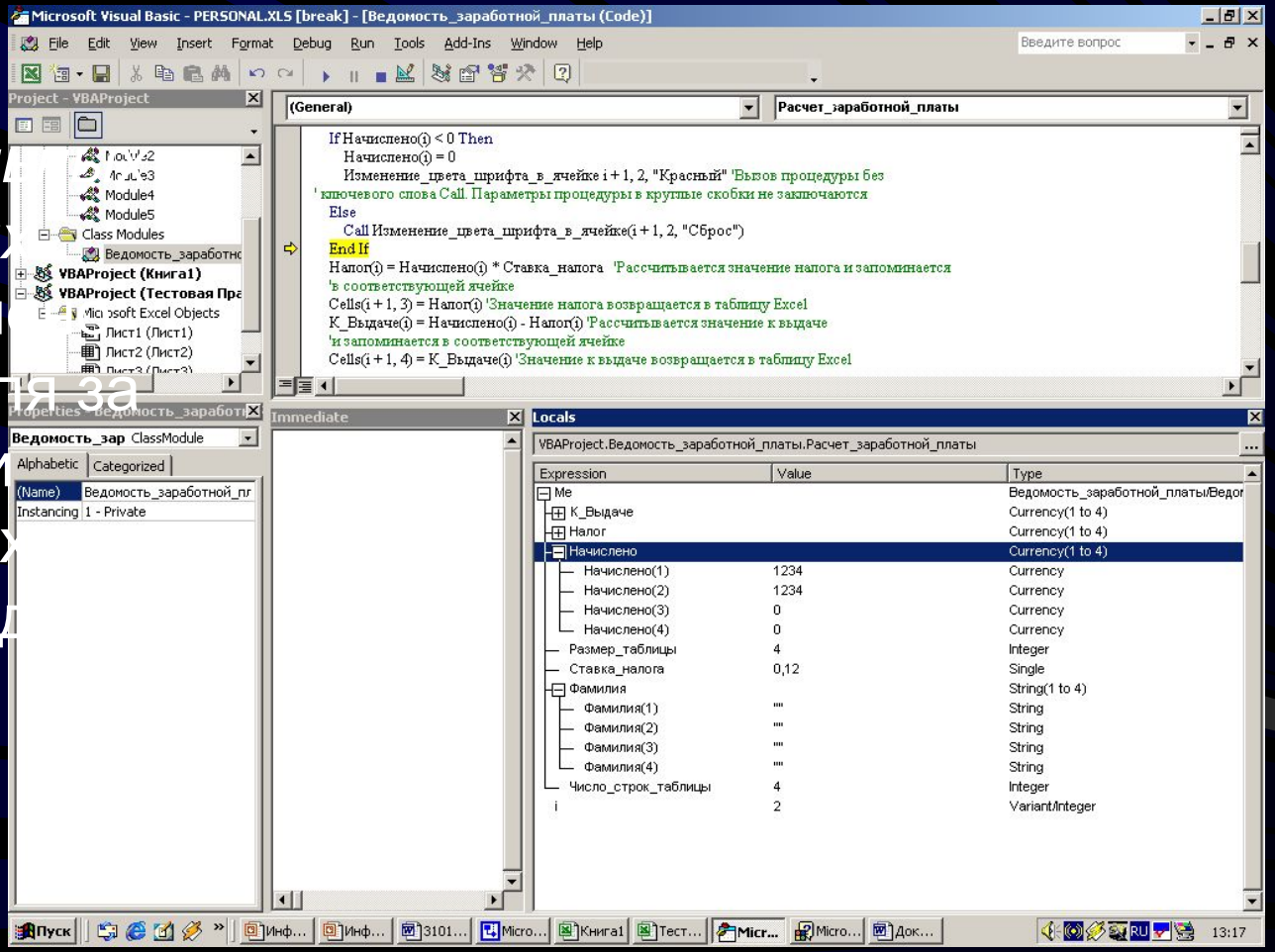
Назначение окон интегрированной среды разработки VBA

- Окно *свойств* перечисляет установленные свойства выбранного объекта



Назначение окон интегрированной среды разработки VBA

Окно *локал*
переменных
предназнач
для контроля за
значениями
переменных
время отлад
программы



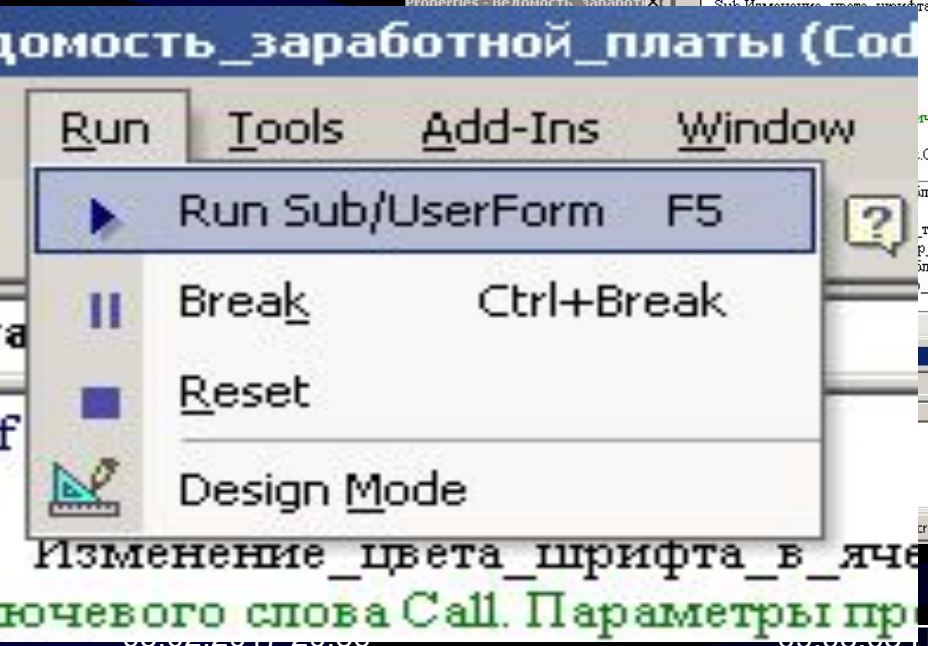
Назначение окон интегрированной среды разработки VBA

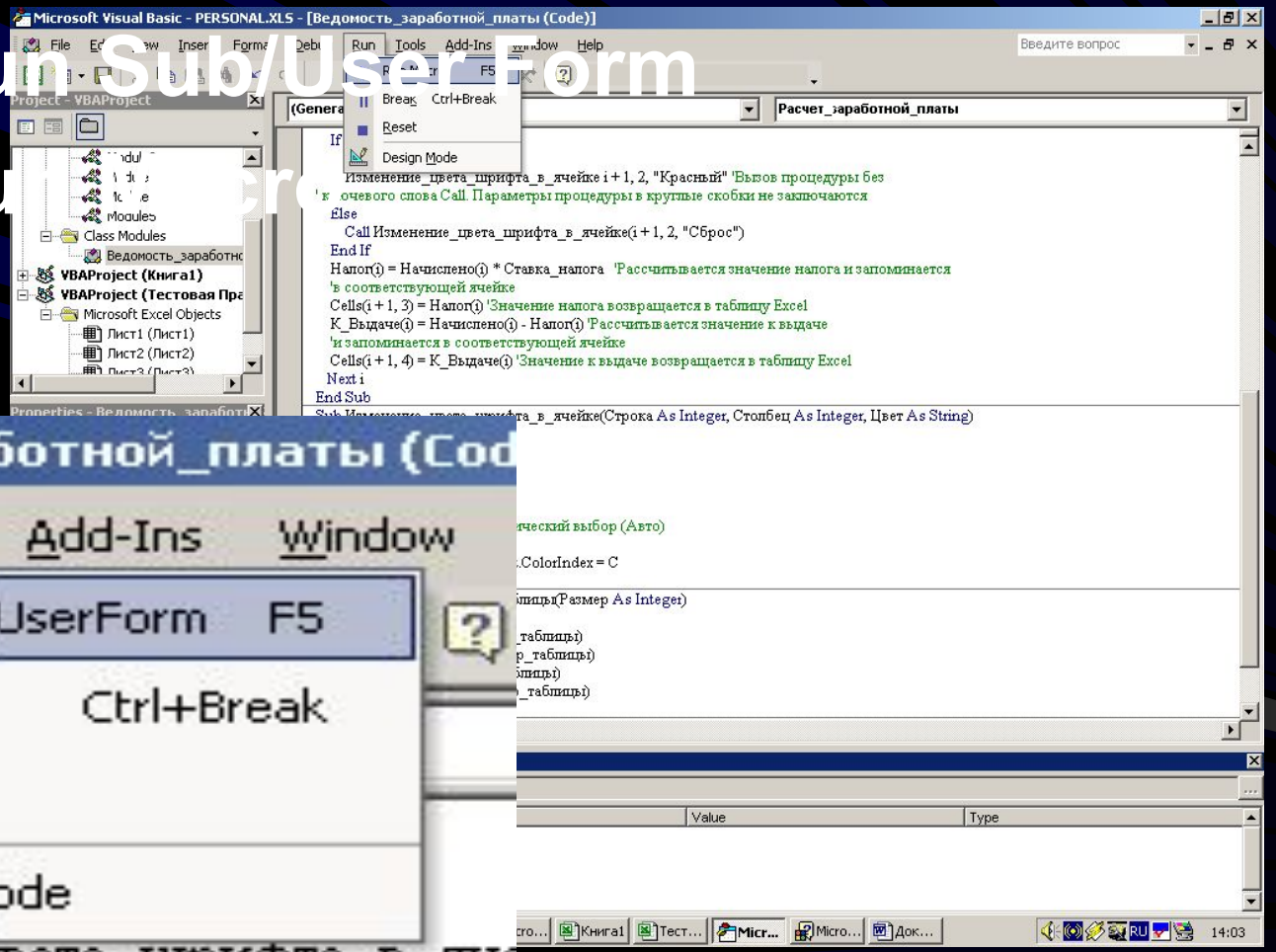
Кроме перечисленных интегрированная среда разработки содержит окна

- *тестирования* **Immediate** (немедленное выполнение), позволяющее изменять значения переменных программы в момент ее выполнения и даже вводить дополнительные операторы;
- *просмотра мгновенных значений* **Watch,** позволяющее вести контроль выбранной переменной программы;
- некоторые другие**.**

1.2.2. Выполнение программы в автоматическом режиме

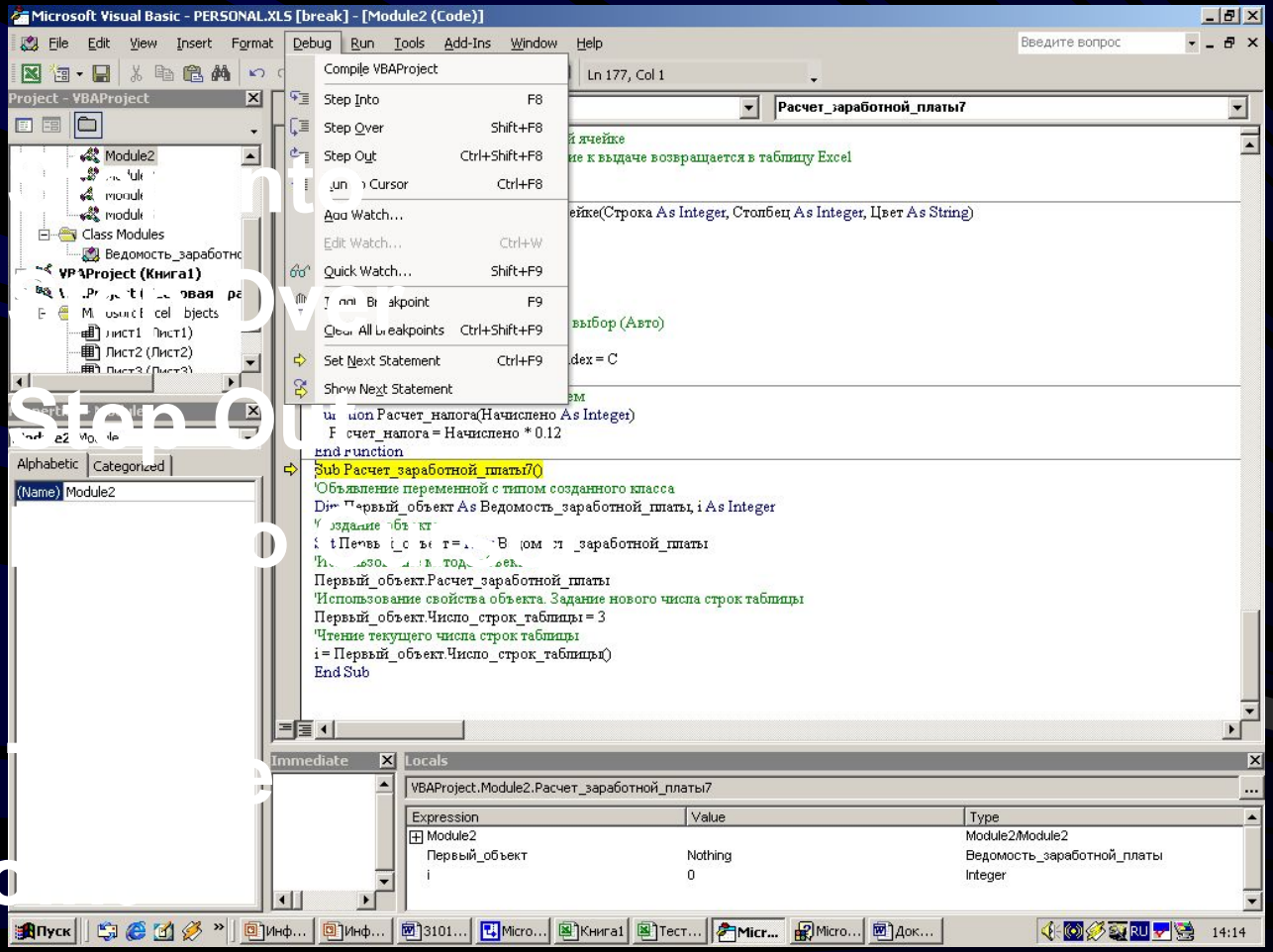
- **Run, Run Sub/UserForm**
- **Run, Ru**





1.2.3. Выполнение программы в режиме отладки

- Debug,
- Debug,
- Debug,
- Debug,
- Debug,
  Breakpo



1.3. Обмен данными между Excel и VBA

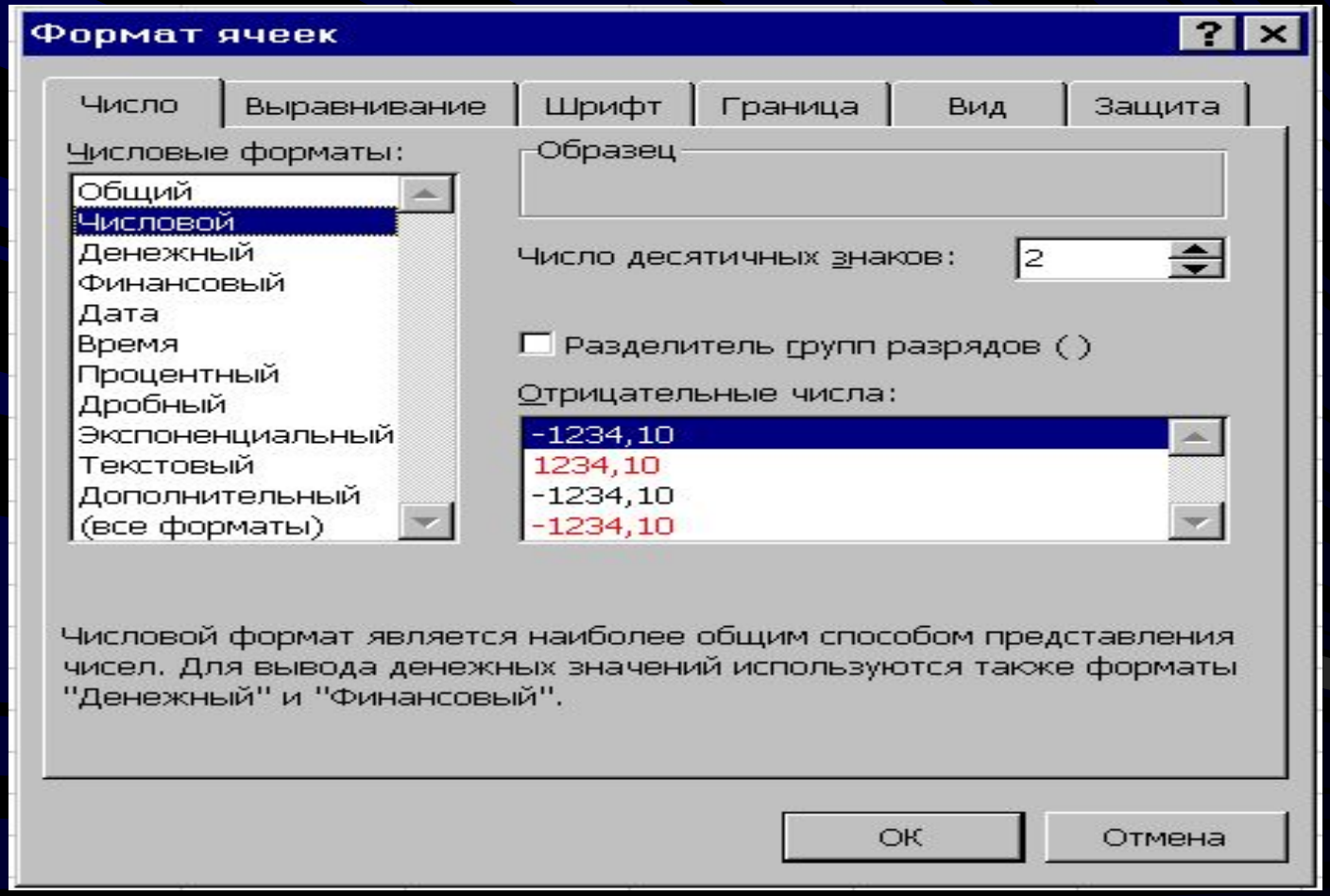
В подразделе рассматривается:

- Типы данных VBA
- Идентификаторы в VBA
- Объявление переменных в VBA
- Массивы в VBA
- Присваивание значения переменной
- Использование свойства Cells() для обмена данными между Excel и VBA

1.3.1. Типы данных VBA

- Тип данных - способ внутреннего представления данных в памяти машины, учитывающий метод их кодирования в одной или нескольких ячейках памяти и предусматривающий возможности их расшифровки или преобразования.

Типы данных VBA



Типы данных Excel

Типы данных VBA

Тип данных	байт		Диапазон значений
Byte (байт)	1		От 0 до 255
Boolean (логический)	2		True или False
Integer (целые)	2	%	От -32768 до 32767
Long (длинное целое)	4	&	От -2147483648 до 2147483647
Single (плавающее обычной точности)	4	!	От -3,402823E38 до -1,401298E-45 и от 1,401298E-45 до 3,402823E38

Типы данных VBA

Тип данных	байт		Диапазон значений
Double (плавающее двойной точности)	8	#	От -1,797693134862З1E308 до -4,940656458412З7E-324 и от 4,940656458412З7E-324 до 1,797693134862З1E308
Currency (денежный)	8	@	От -922337203685477,5808 до 922337203685477,5808
Decimal (масштабируемое целое)	14		+/-79228162514264337593539503З5 и 28 знаков после запятой. Минимальное отличное от нуля значение имеет вид +/-0,0000000000000000000000000001

Типы данных VBA

Тип данных	байт		Диапазон значений
Date (время и дата)	8		От 1 января 100 г. до 31 декабря 9999 г.
Object (объект)	4		Любой указатель объекта
String (строка переменной длины)	10+ длина строки		От 0 до приблизительно двух миллиардов
String (строка постоянной длины)	Длина строки	$	От 1 до 65400

Типы данных VBA

Тип данных	байт		Диапазон значений
Variant (числовые подтипы)	16		От -1,7976931348623 2E308 до -4,94065645841247E-324 и от 4,94065645841247E-324 до 1,7976931348623 2E308
Variant (строковые подтипы)	22+ длина строки		От 0 до приблизительно двух миллиардов
Type (определяемый пользователем)	Определяется элементами типа		Диапазон каждого элемента определяется его типом данных

Типы данных VBA

- Type (определяемый пользователем). Определяется элементами типа. Диапазон каждого элемента определяется его типом данных

```
Type Запись_Ведомости
Фамилия_И_О As String
Начислено_Ведомость As Currency
Налог_Ведомость As Currency
К_выдаче_Ведомость As Currency
End Type
```

1.3.2. Идентификаторы в VBA

- Идентификатором называется символическое имя ячейки памяти.
- Каждый язык программирования содержит свои правила составления таких имен.
- Общим является то, что программист вправе сам придумать имя, что позволяет ему сохранить в нем смысловое значение.

Идентификаторы в VBA

В языке VBA имеются следующие ограничения на имена:

- Длина имени не должна превышать 255 символов.
- Имя должно начинаться с буквы.
- Имя не может содержать точек и символов %, &, !, #, @, $.

Идентификаторы в VBA

В языке VBA имеются следующие ограничения на имена:

- Буквы рассматриваются инвариантно по отношению к регистру, то есть имя Aa и aA есть одно и то же имя.
- Допускается использование символов латыни и кириллицы.
- Совпадения имен идентификаторов с так называемыми *ключевыми* словами не допускается.

Идентификаторы в VBA

- Использование символов кириллицы в именах позволяет программисту создавать осмысленные имена идентификаторов, что облегчает чтение и отладку программы

Идентификаторы в VBA

Возможные варианты идентификаторов языка VBA:

- I, j, Name, Переменная, Результат_вычислений.

Еще варианты записи идентификаторов:

- A%, B&, C!, D#, E@, F$.

В этом случае символы %, &, !, #, @, $ не входят в состав идентификатора и используются в качестве специального признака типа данных

1.3.3. Объявление переменных в VBA

- **Dim I As Integer, Name, j As _ Integer,** Переменная **As _ Integer, GGG As Integer**
- Обратите внимание на то, что если вы не указываете явно тип переменной, то по умолчанию она имеет тип Variant. Так, в рассмотренном выше примере такой тип имеет переменная Name

Объявление переменных в VBA

- Обрабатывая файл исходного текста программы компилятор создает двоичный файл, который впоследствии после дополнительной обработки представляет собой последовательность кодов программы, выполняемой процессором
- Там же в программе отводится область для хранения данных

Объявление переменных в VBA

- Строка объявления переменных обрабатывается компилятором. Результат обработки – резервирование ячеек памяти в двоичном файле
- Адреса ячеек определяются как смещение по отношению к первому байту исполняемой программы
- После загрузки файла в ОЗУ адреса ячеек получают абсолютные значения

1.3.4. Массивы в VBA

- Практика программирования широко использует переменные, обращение к которым ведется как по имени, так и по номеру.
- В этом случае можно говорить о создании переменных табличного типа, когда обращение к данным ведется по имени и номеру (индексу) внутри этого имени.
- Такие переменные обычно называются массивами.

Массивы в VBA

- Массив - последовательно упорядоченные в памяти данные одного типа
- Каждый массив имеет имя
- Имя массива – идентификатор, за которым закреплен свой адрес ОЗУ

Массивы в VBA

- Каждый элемент представляет собой ячейку или последовательность ячеек памяти в зависимости от используемого типа данных
- Количество таких ячеек определяет размер массива

Тип данных	байт		Диапазон значений
Byte (байт)	1		От 0 до 255
Boolean (логический)	2		True или False
Integer (целые)	2	%	От -32768 до 32767
Long (длинное целое)	4	&	От -2147483648 до 2147483647
Single (плавающее обычной точности)	4	!	От -3,402823E38 до -1,401298E-45 и от 1,401298E-45 до 3,402823E38
Double (плавающее двойной точности)	8	#	От -1,79769313486231E308 до -4,94065645841247E-324 и от 4,94065645841247E-324 до 1,79769313486231E308
Currency (денежный)	8	@	От -922337203685477,5808 до 922337203685477,5808
Decimal (масштабируемое целое)	14		+/-79228162514264337593539503З5 и 28 знаков после запятой. Минимальное отличное от нуля значение имеет вид +/-0,0000000000000000000000000001

Массивы в VBA

- Объявления массивов:

Dim YY(25)

Объявляется одномерный массив из 26 элементов. Начальный (базовый) индекс принят по умолчанию равным 0.

Dim ZZ(3,10) As Single

Объявляется двумерный массив ZZ типа Single, первый индекс которого меняется в диапазоне от 0 до 3, а второй в диапазоне от 0 до 10.

Dim SS(-3 To 3,1 To 10) As Integer

Массивы в VBA

- Для обращения к ячейке памяти или элементу массива достаточно в тексте программы использовать соответствующий идентификатор (в случае массива с номером элемента, указанным в скобках).
- В качестве номера элемента массива может выступать не только константа, но и другая переменная, заданная своим идентификатором.
- Недостатком рассмотренного приема является относительно высокая вероятность возникновения ошибки программирования связанной с выходом индекса (номера элемента) за границы массива.

Массивы в VBA

Dim SS(-3 To 3,1 To 10) As Integer

- Обращение к элементу массива в тексте программы с явным указанием номеров элементов: **SS(-2,5)**
- Если переменная **Name** содержит число –2, а ячейка Переменная число 5, то обращение **SS(Name,** Переменная**)** полностью эквивалентно предыдущему.
- Если в процессе предыдущих вычислений переменная **Name** примет значение –4, а мы попытаемся выполнить **SS(Name,** Переменная**)**, то произойдет обращение к несуществующему элементу массива и возникнет ошибка выхода индекса за границы массива.

Массивы в VBA

- Иногда приходится создавать массивы, размер которых невозможно определить на этапе компиляции программы.
- Конечно, можно объявить массивы с запасом, так, чтобы номер максимального элемента массива был заведомо большим максимально возможного числа. Такой прием приводит к нерациональному распределению памяти.
- Альтернативой является метод динамического объявления размера массива. В этом случае конкретный размер массива вычисляется в процессе выполнения программы и память для хранения данных отводится тоже во время выполнения.

Массивы в VBA

Dim Начислено**() As Currency, i As Integer**

i = 10

ReDim Начислено**(1 To i)**

- Массив Начислено**()** первоначально был объявлен как массив неопределенной длины. Инструкция **ReDim** изменила массив, причем память под него была отведена в момент выполнения программы.

1.3.5. Присваивание значения переменной

- Оператор присваивания обеспечивает занесение информации в ячейки памяти, связанные с идентификатором и имеет символ равенства (=).

i = 10

- В отличие обычного равенства, которое выполняется всегда, оператор присваивания имеет динамические свойства (зависит от времени).

Оператор присваивания

- При выполнении оператора присваивания результат вычислений правой части оператора заносится в ячейку памяти, указанную слева от знака равенства.
- Содержимое ячейки, указанной слева от символа =, имело одно значение до выполнения оператора и другое после его выполнения.
- Задавая последовательность операторов присваивания мы можем программировать запись данных в ячейки памяти ЭВМ.

1.3.6. Использование свойства Cells() для обмена данными между Excel и VBA

- Отдельную проблему представляет прямая и обратная передача данных из таблицы Excel в ячейки памяти, объявленные в программе, написанной на VBA.
- Автоматически созданный макрос непосредственно манипулирует с ячейками таблицы используя стили ссылки на ячейки в Excel: A1 и R1C1.
- Такой прием может быть использован и в рабочей программе, однако в этом случае ее модификация и использование существенно затруднены.

Использование свойства Cells() для обмена данными между Excel и VBA

- Гораздо предпочтительнее использовать свойство Cells() стандартного объекта Excel Range.
- Сам объект представляет собой ячейку, столбец, строку или выделенный диапазон листа Excel.
- Свойство Cells() позволяет непосредственно обратиться к объекту Excel по номеру строки и колонки.
- Поскольку это свойство установлено по умолчанию для рабочего листа Excel, то его можно использовать без дополнительных указаний.

Использование свойства Cells() для обмена данными между Excel и VBA

```
Dim ddd, x, y
ddd= Cells(3,7)
X=3
Y=7
 Cells(x,y)=“Сумма”
```

Использование свойства Cells() для обмена данными между Excel и VBA

- Если запись свойства стоит слева от оператора присваивания, то производится запись данных в ячейку таблицы, если справа, то считывание значения из ячейки таблицы.
- Кроме собственно записи данных свойство Cells() в сочетании со свойствами других объектов (Font, Color и т.п.) позволяет задавать параметры шрифта, его цвет, фон и так далее.

Использование свойства Cells() для обмена данными между Excel и VBA

- Для изучения этих возможностей целесообразно ознакомиться с описанием соответствующих свойств и объектов в литературе, воспользоваться Help-системой или, что проще всего, запустить режим записи макроса в Excel, выполнить, например, установку цвета и изучить текст полученного макроса.

2. Операции и операторы VBA

В разделе рассматривается:

- Операции VBA
- Операторы VBA

2.1 Операции VBA

В подразделе рассматривается:

- Арифметические операции
- Операции сравнения
- Логические операции
- Операции со строками

2.1.1. Арифметические операции

Операции	Приоритет	Название	Пример	Результат
Арифметические операции			A=11 B=5	
-	3	Смена знака	-A	-11
+	7	Сложение	A+B	16
-	7	Вычитание	A-B	6
*	4	Умножение	A*B	55
/	4	Деление	A/B	2.2
\	5	Целочисленное деление	A\B	2
Mod	6	Остаток от деления по модулю	A Mod B	1
^	2	Возведение в степень	A^B	161015

2.1.2. Операции сравнения

Операции	Приоритет	Название	Пример	Результат
Операции сравнения			A=11 B=5	
<	8	Меньше	A<B	False
>	8	Больше	A>B	True
<=	8	Меньше и равно	A<=B	False
>=	8	Больше и равно	A>=B	True
<>	8	Не равно	A<>B	True
=	8	Равно	A=B	False
Is		Сравнение со ссылкой на объекты	Dim A,B,C,D,E Set A=D Set B=D Set C=E F=A Is B F=A Is C	 True False

2.1.3. Логические операции

Операции	Приоритет	Название	Пример	Результат
Логические операции			A=True B=True C=False D=False	
And	10	Логическое умножение (и)	A And B A And C C And B C And D	True False False False
Or	11	Логическое сложение (или)	A Or B A Or C C Or B C Or D	True True True False
Xor	12	Исключающее или	A Xor B A Xor C C Xor B C Xor D	False True True False
Not	9	Отрицание	E=Not B E=Not D	False True
Imp	14	Импликация	A Imp B A Imp C C Imp B C Imp D	True False True True
Eqv	13	Эквивалентность	A Eqv B A Eqv C C Eqv B C Eqv D	True False False True

2.1.4. Операции со строками

Операции	Приоритет	Название	Пример	Результат
Операции со строками			A=”abc” B=”123”	
&		Сцепление строк	A&B	”abc123”
Like		Сравнение строк		

2.2. Операторы VBA

В подразделе рассматривается:

- Правила записи операторов в языке VBA
- Оператор присваивания Let
- Условный оператор
- Оператор ветвления
- Семейство операторов For
- Семейство операторов Do

Операторы VBA

- Оператором называется самостоятельная конструкция языка программирования, которая может быть отдельно откомпилирована и выполнена в виде заранее определенной последовательности кодов процессора

2.2.1. Правила записи операторов в языке VBA

- Операторы записываются на отдельных строчках и могут не нумероваться.
- Для размещения нескольких операторов на одной строке между ними необходимо поставить символ двоеточие (:). Этот же символ используется для обозначения меток.

Правила записи операторов в языке VBA

- Для переноса продолжения оператора на следующую строку используется комбинация символов пробел знак подчеркивания (_). Нельзя разбивать переносом выражения и строки. Допускается не более семи переносов строк одного оператора.

2.2.2. Оператор присваивания Let

- Оператор присваивания **Let** в VBA в момент выполнения записывает в переменную, указанную слева от символа равенства некое значение, указанное справа от символа равенства, результат вычисления функции и т.п.
- Формат оператора

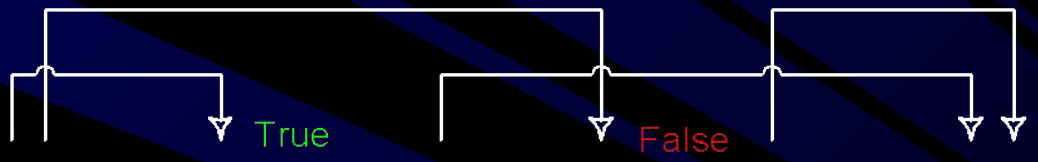
  Let Переменная=Значение

Оператор присваивания Let

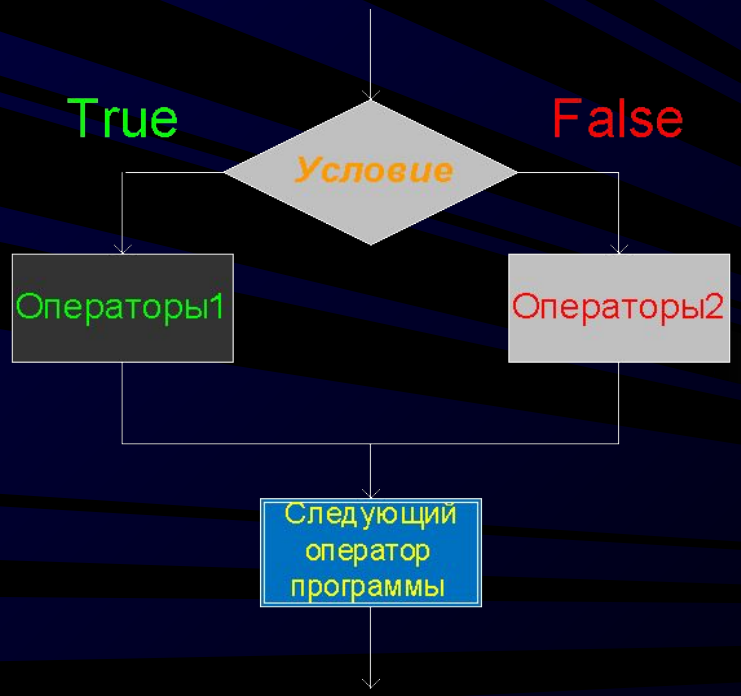
- Существует сокращенная форма записи оператора **Let** при которой ключевое слово опускается и сохраняется только символ равенства
- Сокращенный формат оператора

  Переменная=Значение

2.2.3. Условный оператор



If ***Условие*** Then Операторы1 Else: Операторы2 Endif Следующий оператор программы



Условный оператор

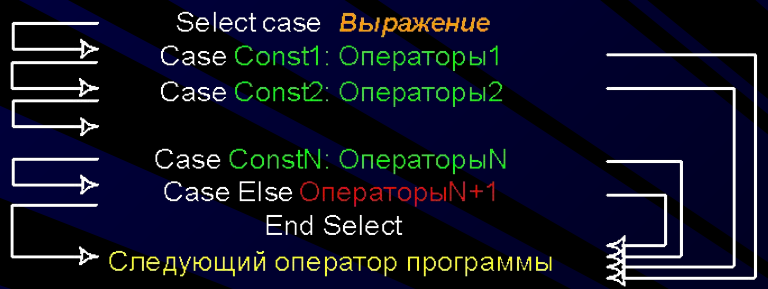
- Формат условного оператора

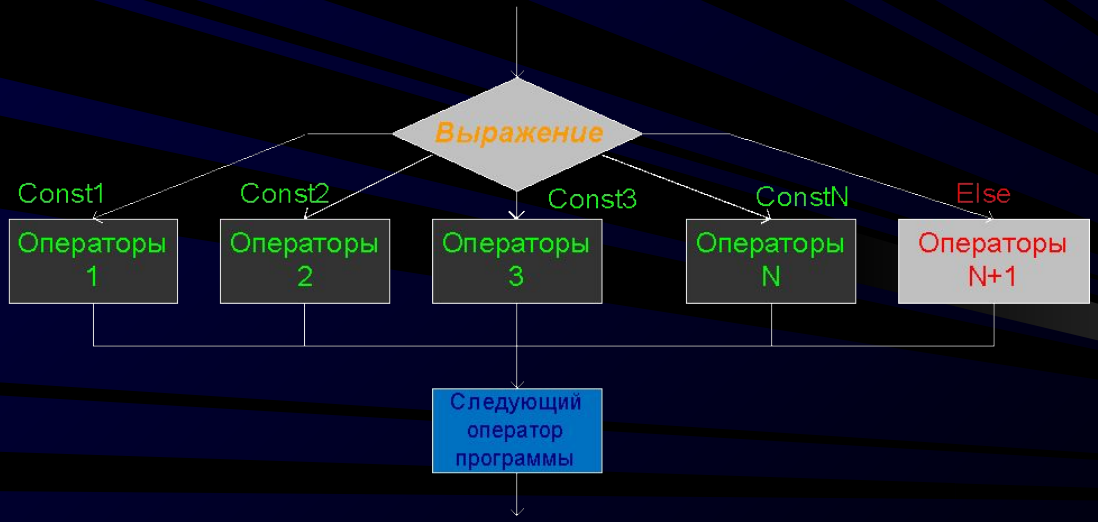
If Условие **Then** [Операторы] [**Else** Операторы_Else] **End If**

Пример программы с условным оператором

```
Dim Таблица(10), i, extr
i = 5
extr = -20
If Таблица(i) > extr Then
    extr = Таблица(i)
Else
End If
```

2.2.4. Оператор ветвления





Оператор ветвления

- Формат оператора ветвления

Select Case Выражение

[**Case** Значение1 [Операторы1]]

[**Case** ЗначениеN [ОператорыN]]

[**Case Else** [ОператорыElse]]

End Select

Оператор ветвления

- Пример программы с оператором ветвления

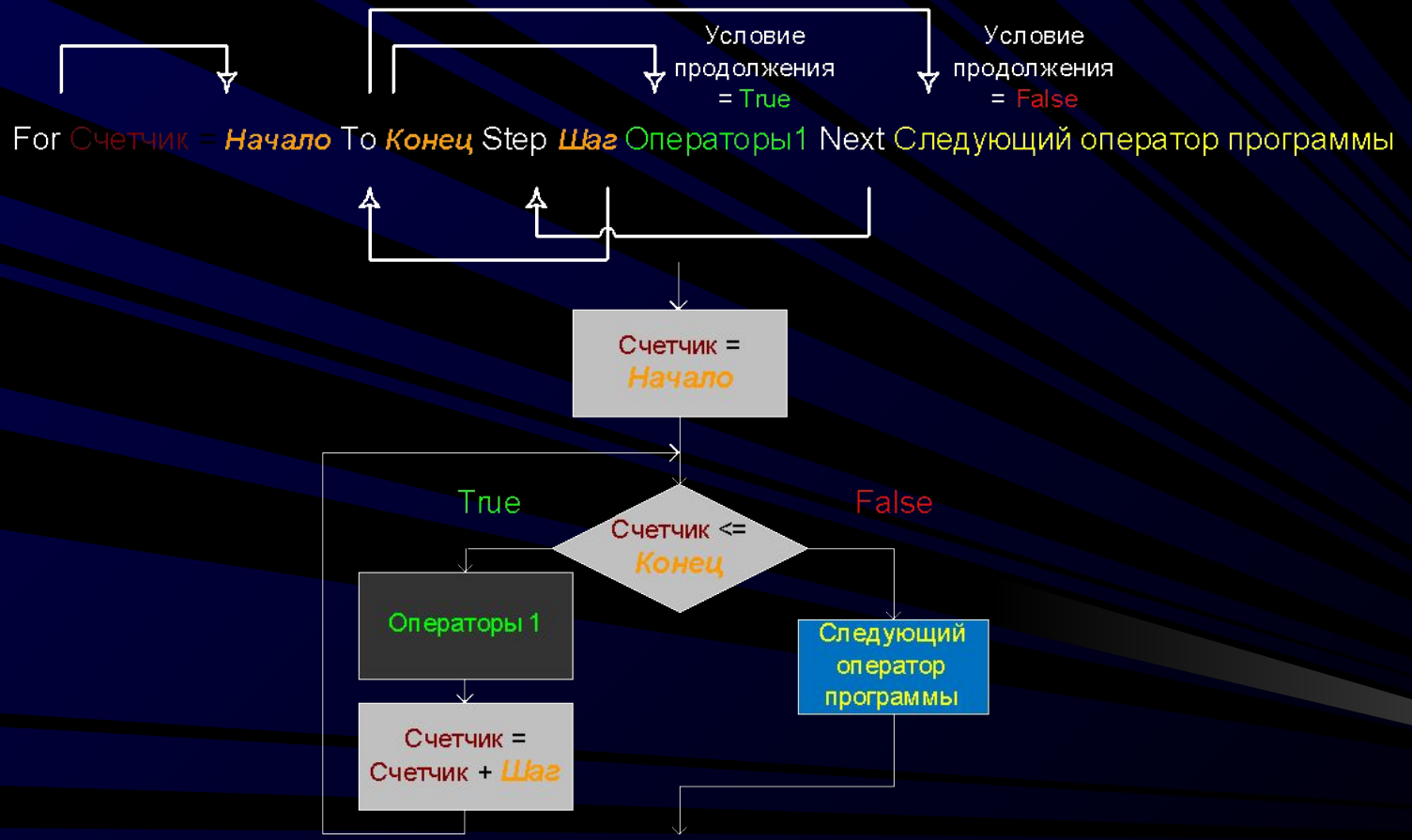
```
Dim РежимРаботы As String, День As Integer
День = 2
Select Case День
    Case 1
    РежимРаботы = "Прием документов"
    Case 2, 3, 4
    РежимРаботы = "Работа с документами"
    Case 5
    РежимРаботы = "Выдача документов"
    Case 6, 7
    РежимРаботы = "Выходные дни"
    Case Else
    РежимРаботы = "Ошибка задания номера дня"
End Select
```

2.2.5. Операторы цикла

В пункте рассматривается:

- Семейство операторов For
- Семейство операторов Do

2.2.5.1. Семейство операторов For



Семейство операторов For

Форматы оператора

For Счетчик=Начало **To** Конец [**Step** Шаг]
[Операторы]
[Exit For]
[Операторы]
Next [Счетчик]

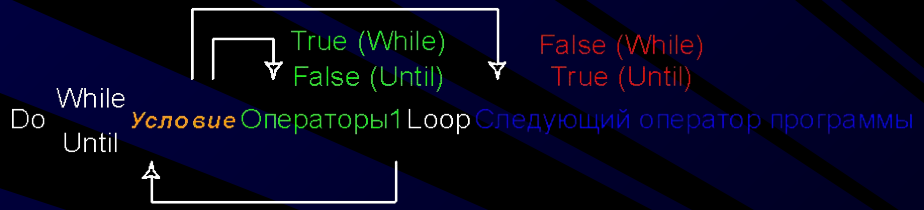
For Each Элемент **In** Группа
[Операторы]
[Exit For]
[Операторы]
Next [Элемент]

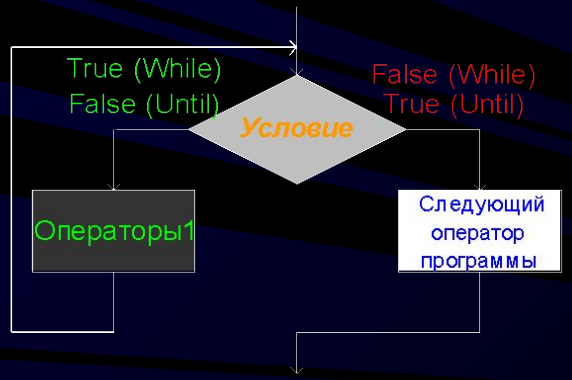
Семейство операторов For

Пример программы с оператором цикла for

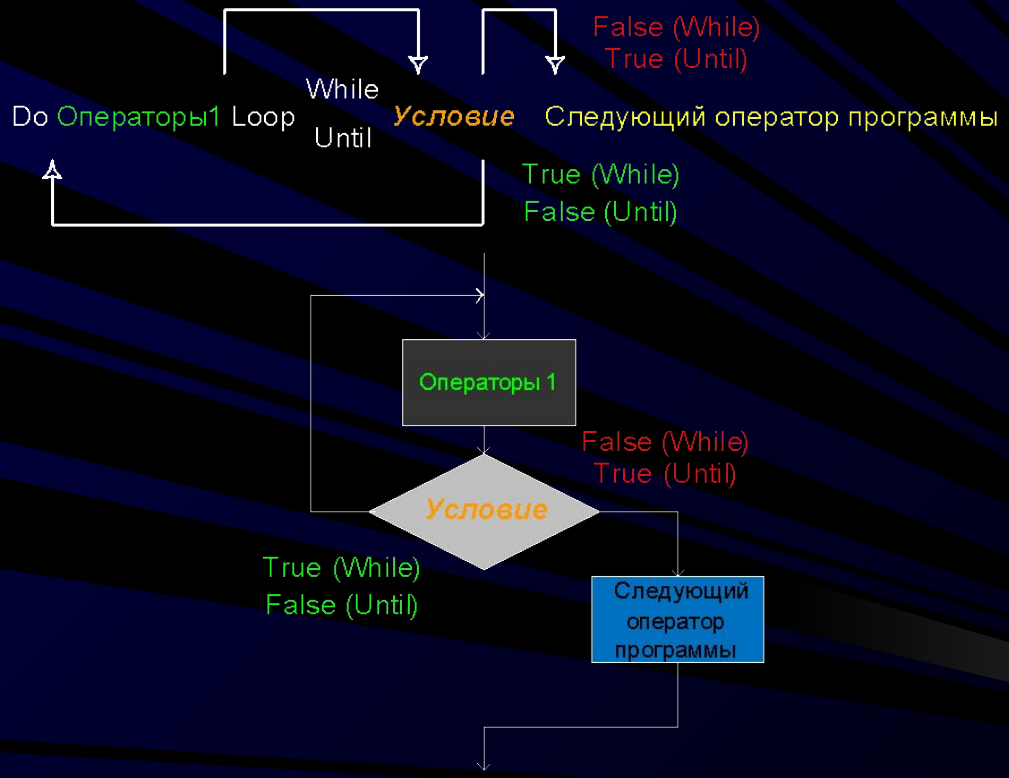
```
Dim i As Integer, AA(10) As Double, _
BB(10) As Double,  j As Variant
For i = 1 To 10
  AA(i) = i
Next i
For Each j In AA
  BB(j) = AA(j)
Next j
```

2.2.5.2. Семейство операторов Do





Семейство операторов Do



Семейство операторов Do

- Операторы While выполняются до тех пор, пока Условие = True.

Do [**While** Условие]
  [Операторы]
  [Exit Do]
  [Операторы]
Loop

или

Do
  [Операторы]
  [Exit Do]
  [Операторы]
Loop [**While** Условие]

Семейство операторов Do

Пример программы с оператором цикла While

```
Dim i As Integer, AA(10) As Double, _
BB(10) As Double, j As Variant
i = 1
    Do While i <= 10
        AA(i) = i
        i = i + 1
     Loop
     j = 1
     Do
        BB(j) = AA(j)
        j = j + 1
     Loop While j <= 10
```

Семейство операторов Do

- Операторы Until выполняются до тех пор, пока Условие = False

Do [**Until** Условие]
[Операторы]
[Exit Do]
[Операторы]
Loop

или

Do
[Операторы]
[Exit Do]
[Операторы]
Loop [**Until** Условие]

Семейство операторов Do

Пример программы с оператором цикла Until

```
Dim i As Integer, AA(10) As Double, _
 BB(10) As Double, j As Variant
i = 1
Do Until i > 10
    AA(i) = i
    i = i + 1
Loop
j = 1
Do
    BB(j) = AA(j)
    j = j + 1
Loop Until j > 10
```

Семейство операторов Do

- Оператор выполняется, пока Условие = True

While Условие

[Операторы]

Wend

3. Процедуры и функции

В подразделе рассматривается:

- Процедуры в VBA
- Функции в VBA
- Возврат значений из процедур и функций в вызывающую программу через список формальных параметров
- Встроенные функции VBA
- Функции Excel, определенные пользователем

3.1. Процедуры в VBA

- *Процедурой* называется самостоятельная программа, предназначенная для решения определенной задачи.
- Каждая процедура имеет имя. Это имя является идентификатором процедуры.
- Макрос Excel представляет собой процедуру VBA.

Процедуры в VBA

- Каждая процедура может быть вызвана по имени. Если вызов отсутствует, то процедура выполняться не будет.
- Процедура может быть запущена на выполнения с помощью интегрированной среды отладки VBA.
- Каждая процедура имеет свои коды, которые должны быть оформлены заданным языком программирования способом.
- Для решения задачи процедура может потребовать набор аргументов (исходные данные), которые передаются ей в момент вызова.

Процедуры в VBA

Формат описания процедуры:

[**Private** или **Public**] [**Static**] **Sub** Имя [(СписокАргументов)]

[Операторы]

[Exit Sub]

[Операторы]

End Sub

Процедуры в VBA

Формат описания списка аргументов:

[**Optional**] [**ByVal** или **ByRef**] [**ParamArray**] ИмяПеременной[**()**] [**As** Тип] [=ПоУмолчанию]

Optional - необязательный элемент. Должен иметь тип Variant. Все последующие элементы списка должны иметь такой же ключ и тип.

Процедуры в VBA

Формат описания списка аргументов:

[**Optional**] [**ByVal** или **ByRef**] [**ParamArray**] ИмяПеременной[**()**] [**As Тип**] [=поУмолчанию]

Способ передачи параметров

ByVal –по значению, ByRef – по ссылке (по умолчанию)

Процедуры в VBA

Формат описания списка аргументов:

[**Optional**] [**ByVal** или **ByRef**]
[**ParamArray**] ИмяПеременной[**()**] [**As** Тип]
[**=**поУмолчанию]

ParamArray – неизвестное число параметров. Может быть использовано только с последним элементом списка формальных параметров и позволяет передавать динамически объявляемый массив

Процедуры в VBA

Формат описания списка аргументов:

[**Optional**] [**ByVal** или **ByRef**]
[**ParamArray**] ИмяПеременной[**()**] [**As** Тип]
[**=**поУмолчанию]

ИмяПеременной[()] - аргумент процедуры (обычная переменная или массив). Идентификатор, имеющий смысл формального параметра процедуры. Может быть несколько аргументов.

Процедуры в VBA

Формат описания списка аргументов:

[**Optional**] [**ByVal** или **ByRef**] [**ParamArray**] ИмяПеременной[**()**] [**As** Тип] [**=**поУмолчанию]

[As Тип] - тип данных аргумента

Процедуры в VBA

Формат описания списка аргументов:

[**Optional**] [**ByVal** или **ByRef**]
[**ParamArray**] ИмяПеременной[**()**] [**As** Тип]
[**=**поУмолчанию]

[=поУмолчанию] - значение аргумента по умолчанию

Процедуры в VBA

- После заголовка процедуры следует конечное число обычных операторов языка VBA, представляющих собой тело определения функции. Если в их состав входят операторы объявления переменных **Dim**, то имеет место объявление собственных локальных переменных процедуры.
- Если в заголовке процедуры не указан ключ **Static**, то эти переменные не сохраняют свои значения между вызовами и каждый раз значения в них должны записываться заново.

Процедуры в VBA

Пример процедуры:

```
Sub ПримерПроцедуры(ByVal День As  Integer, _
ByRef РежимРаботы As String)
    Select Case День
     Case 1
    РежимРаботы = "Прием документов"
     Case 2, 3, 4, 5
    РежимРаботы = "Выдача документов"
     Case Else
    РежимРаботы = "Выходные дни"
    End Select
End Sub
```

Процедуры в VBA

Пример вызывающей процедуры:

```
Sub ДемонстрацияПримераВызоваПроцедуры()
Dim a As Integer, b As Integer , _
 s As String, ss As String
a = 1
Call ПримерПроцедуры(a, s)
b = 6
ПримерПроцедуры b, ss
End Sub
```

Процедуры в VBA

- Аргументами процедуры в момент ее описания являются так называемые *формальные параметры*. Они используются как полноправные участники любых операций и операторов тела процедуры для указания необходимой последовательности действий
- *Формальные параметры* получают физические адреса памяти для своего размещения они только в момент вызова процедуры. Обычно говорят, что процедура (функция) вызывается с *фактическими параметрами*
- Использование формальных параметров позволяет многократно вызывать процедуру из разных точек программы с различными аргументами.

3.2. Функции в VBA

- *Функцией* называется вызываемая через оператор присваивания самостоятельная программа, предназначенная для решения определенной задачи.

Функции в VBA

[**Public** или **Private**] [**Static**] **Function** Имя [**(**СписокАргументов**)**] [**As** Тип]
    [Операторы]
    [Имя=Выражение]
    [Exit Function]
    [Операторы]
    [Имя=Выражение]
End Function

Функции в VBA

Формат описания списка аргументов:

[**Optional**] [**ByVal** или **ByRef**]
[**ParamArray**] ИмяПеременной[**()**] [**As** Тип]
[**=**поУмолчанию]

Функции в VBA

```
Function ПримерФункции(ByVal День As Integer) As String
Dim РежимРаботы As String
    Select Case День
    Case 1
    РежимРаботы = "Прием документов"
    Case 2, 3, 4, 5
    РежимРаботы = "Выдача документов"
    Case Else
    РежимРаботы = "Выходные дни"
    End Select
ПримерФункции=РежимРаботы
End Function


 Sub ДемонстрацияПримераВызоваФункции()
    Dim a As Integer, s As String
    a = 1
    S = ПримерФункции (a)
    b = 6
 End Sub
```

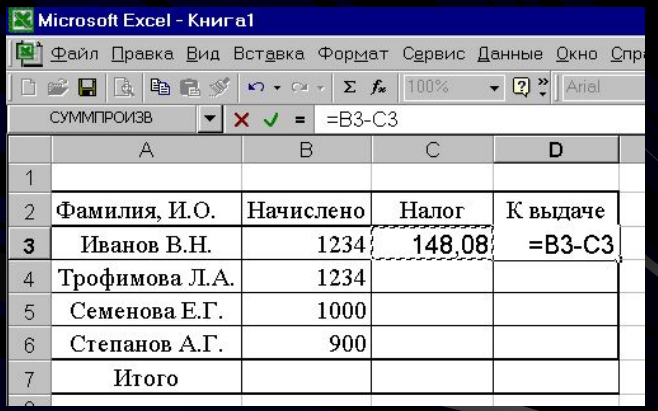
3.3. Возврат значений из процедур и функций в вызывающую программу через список формальных параметров

- С помощью одной функции можно рассчитать и передать в вызывающую процедуру, например, сразу два значения. Одно значение передается в точку вызова обычным способом. Другое значение изменяет формальный параметр и позднее может быть использовано вызывающей программой

Возврат значений из процедур и функций в вызывающую программу через список формальных параметров

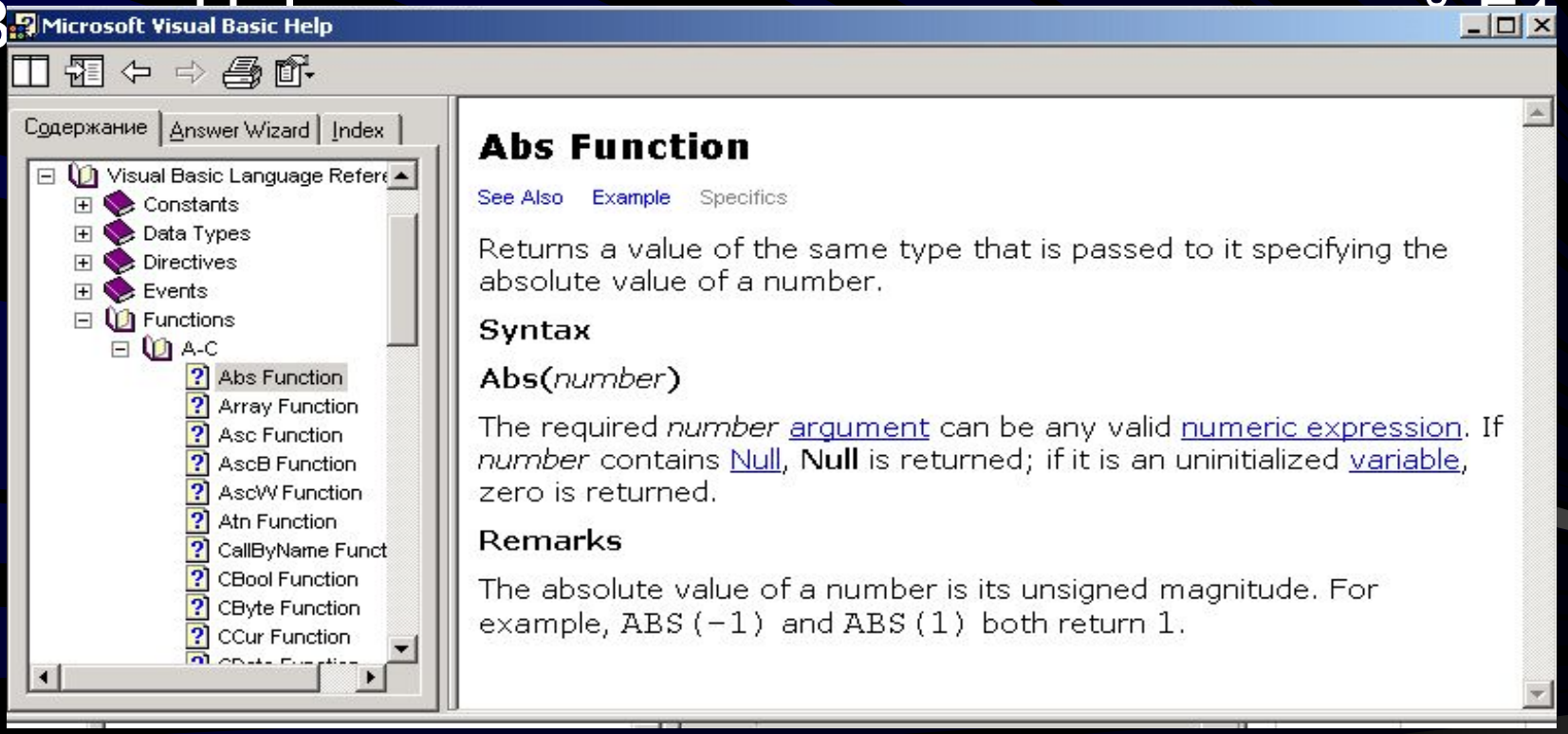
```
Option Explicit
Type Запись_Ведомости  'Определение типа данных, заданного пользователем
Фамилия_И_О As String
Начислено_Ведомость As Currency
Налог_Ведомость As Currency
К_выдаче_Ведомость As Currency
End Type
'.....
' Фрагмент программы вызова рассматриваемой функции Расчет_Зарплаты

Dim Ведомость(4) As Запись_Ведомости , j As Integer  'Объявление массива структур
j = 3
Ведомость(j).Начислено_Ведомость = 1000
 'Вызов функции Расчет_Зарплаты
Ведомость(j).К_выдаче_Ведомость = _
Расчет_Зарплаты(Ведомость(j).Начислено_Ведомость, 0.12,  _
Ведомость(j).Налог_Ведомость)
'После завершения работы функции ячейка Ведомость(j).К_выдаче_Ведомость
'содержит результаты расчета суммы к выдаче, а в ячейке
'Ведомость(j).Налог_Ведомость находятся результаты расчета величины
'подоходного налога
'.....
Private Function Расчет_Зарплаты(ByVal Начислено As _
      Currency, ByVal Ставка_налога As Double, _
      ByRef Подоходный_налог As Currency) As Currency
'Формальный параметр Подоходный_налог также используется для возврата результатов вычислений в вызывающую программу.
       Подоходный_налог = Начислено * Ставка_налога
      Расчет_Зарплаты = Начислено - Подоходный_налог
End Function
```

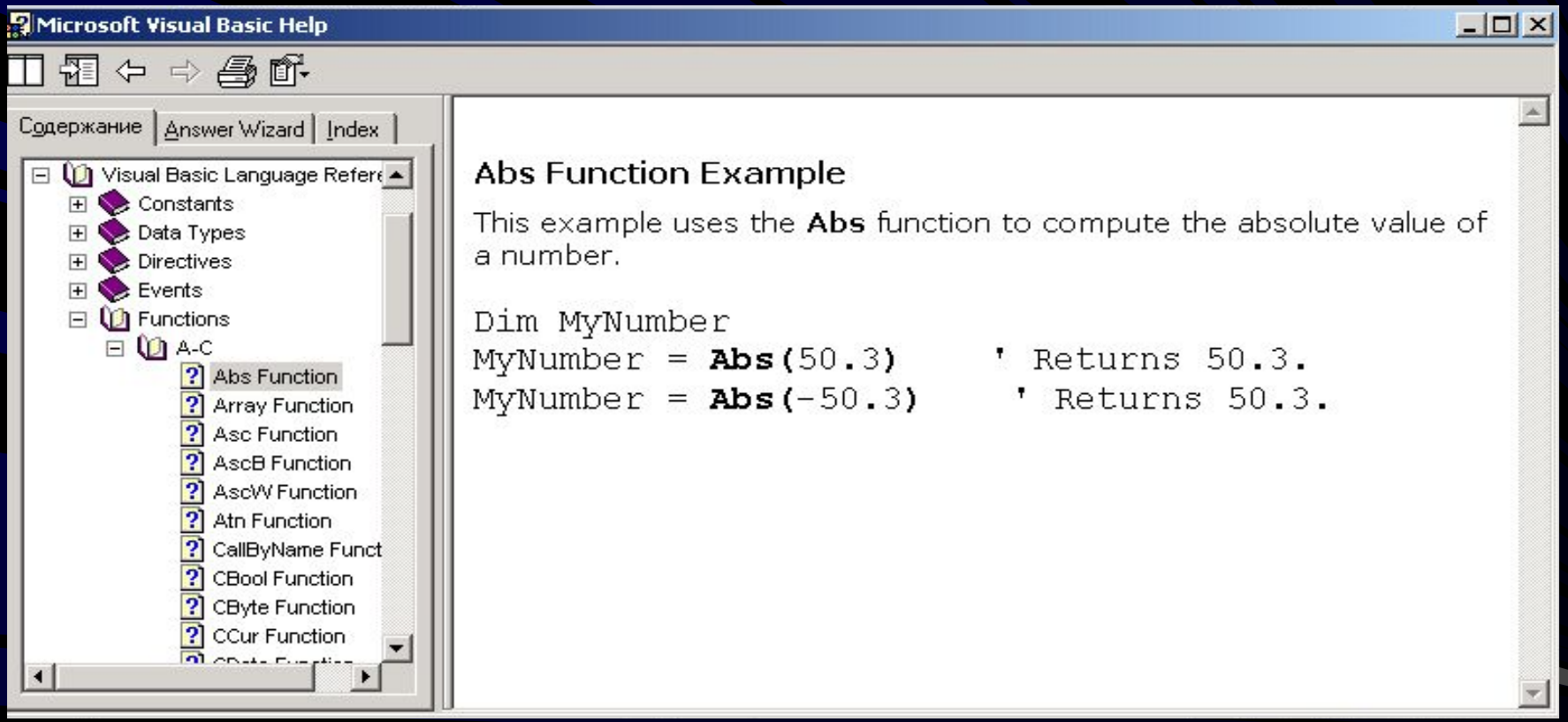


3.4. Встроенные функции VBA

- Перечень встроенных функций VBA приведен в Help-системе
- В



Встроенные функции VBA



3.5. Функции Excel, определенные пользователем

Функция, определенная пользователем

```
Public Function Расчет_налога(Начислено As Integer)
  Расчет_налога = Начислено * 0.12
End Function
```

4. Классы и объекты в языке VBA

В подразделе рассматривается:

- Объекты и классы как конструкции языка VBA
- Создание пользовательского класса
- Создание объекта на основе класса

4.1. Объекты и классы как конструкции языка VBA

- Объектно-ориентированное программирование - технология программирования, при которой программа рассматривается как набор дискретных объектов, содержащих, в свою очередь, наборы структур данных и процедур, взаимодействующих с другими объектами
- Класс в программировании – множество объектов с одинаковой структурой, поведением и отношением к объектам других классов

Объекты и классы как конструкции языка VBA

- Класс – это определенный пользователем тип данных
- Объект – это экземпляр класса содержащий определенные данные
- Класс содержит описание структуры объекта и ограниченный набор функций и процедур, описывающих свойства и поведение объектов
- Память для хранения набора данных объекта резервируется в момент создания объекта и освобождается вместе с его удалением

Объекты и классы как конструкции языка VBA

- Использование технологии классов и объектов позволяет найти компромисс между потребностями программирования в использовании глобальных и статических переменных и требованиями обеспечения надежности программирования

Объекты и классы как конструкции языка VBA

- Кроме указанного, у программиста появляется ряд дополнительных возможностей, которые позволяют по новому взглянуть на методы программирования сложных задач

Объекты и классы как конструкции языка VBA

- Пользовательские классы могут быть созданы непосредственно программистом
- Библиотечные классы описаны в Help – системе или в литературе. Каждая программная система пакета Microsoft Office имеет собственный набор библиотечных классов

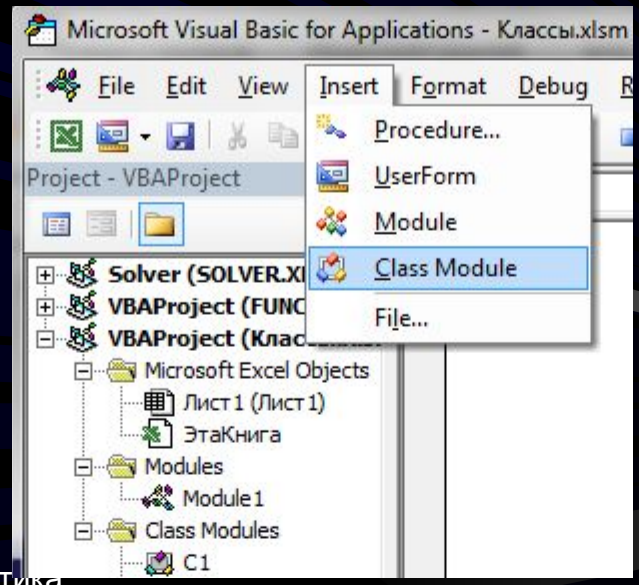
4.2. Создание пользовательского класса

Структура подраздела:

- Общая структура модуля класса
- Описание переменных класса
- Описание свойств класса
- Описание методов класса
- Описание процедур создания и удаления объектов класса
- Библиотечные классы VBA и связанные с ними события

4.2.1. Общая структура модуля класса

- Для создания класса используется специальный модуль проекта VBA, который называется модуль класса (Class Modules). Он создается командой Вставка (Insert) главного меню VBA



Общая структура модуля класса

- Имя модуля класса совпадает с именем создаваемого класса
- В программе может использоваться несколько классов

Общая структура модуля класса

- Типовая структура содержания модуля класса имеет следующий вид:
  - блок описания переменных класса;
  - блок описания процедур создания и удаления объектов класса;
  - блок описания свойств класса;
  - блок описания методов класса.

Общая структура модуля класса

Модуль **класса**

Class Modules

Блок описания переменных класса

Dim

Блок определения процедур создания и удаления объектов класса

Sub Class_Initialize()
Sub Class_Terminate ()

Блок определения свойств класса

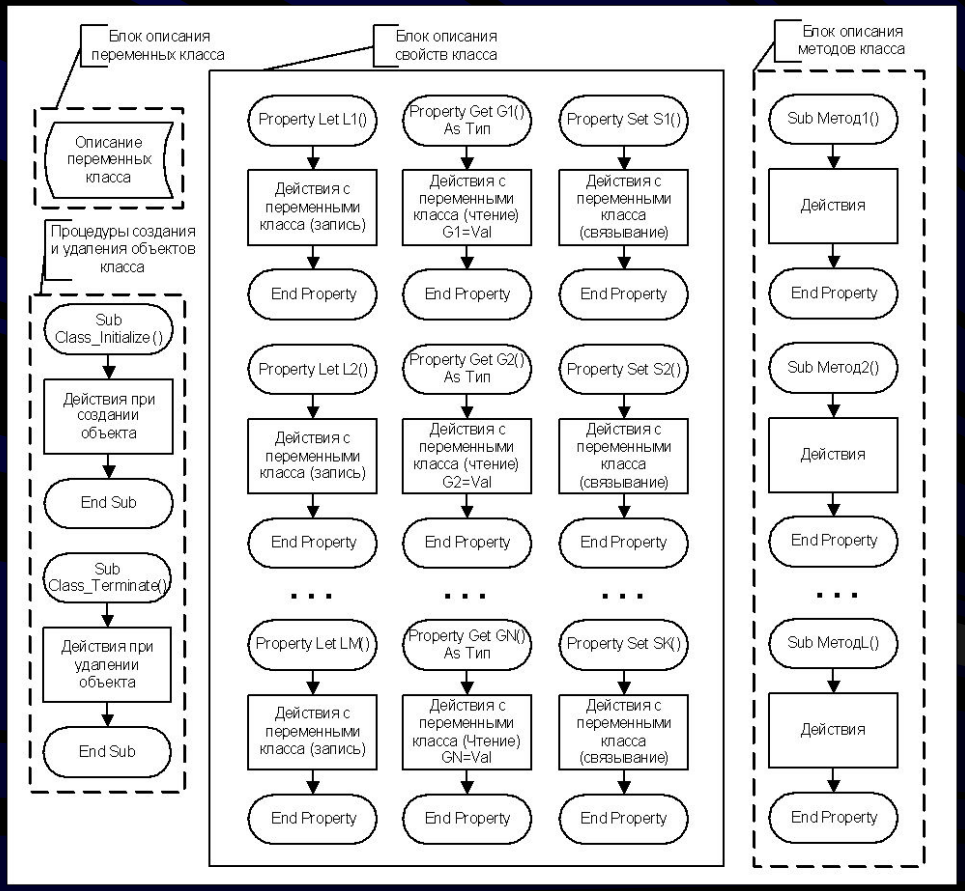
Property Let Имя [(Список_аргументов)]
Property Get Имя [(Список_аргументов)] As Тип
Property Set Имя [(Список_аргументов)]

Блок определения методов класса

Sub Имя [(Список_аргументов)]

Типовая структура содержания модуля класса

Общая структура модуля класса

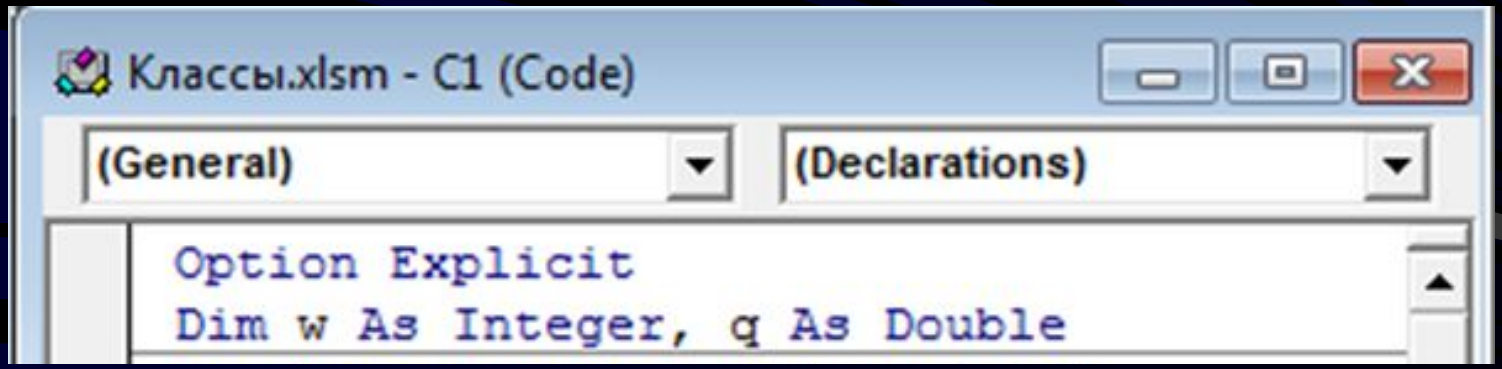


Общая структура модуля класса

- Модуль класса содержит описание структуры данных объекта и коды общих для всех объектов функций и процедур

4.2.2. Описание переменных класса

- Для описания переменных используется обычный оператор **Dim**
- Используется синтаксис и правила объявления переменных языка VBA



Описание переменных класса

- Основное отличие объявления переменных класса от переменных обычной программы заключается в том, что в момент объявления под них не резервируется память
- Резервирование памяти под переменные класса происходит только в момент создания объекта

Описание переменных класса

- Если на основе модуля класса создается несколько объектов, то для каждого резервируется свой набор ячеек памяти, характеризующих его состояние. Поэтому состояние одного объекта не зависит от состояния другого

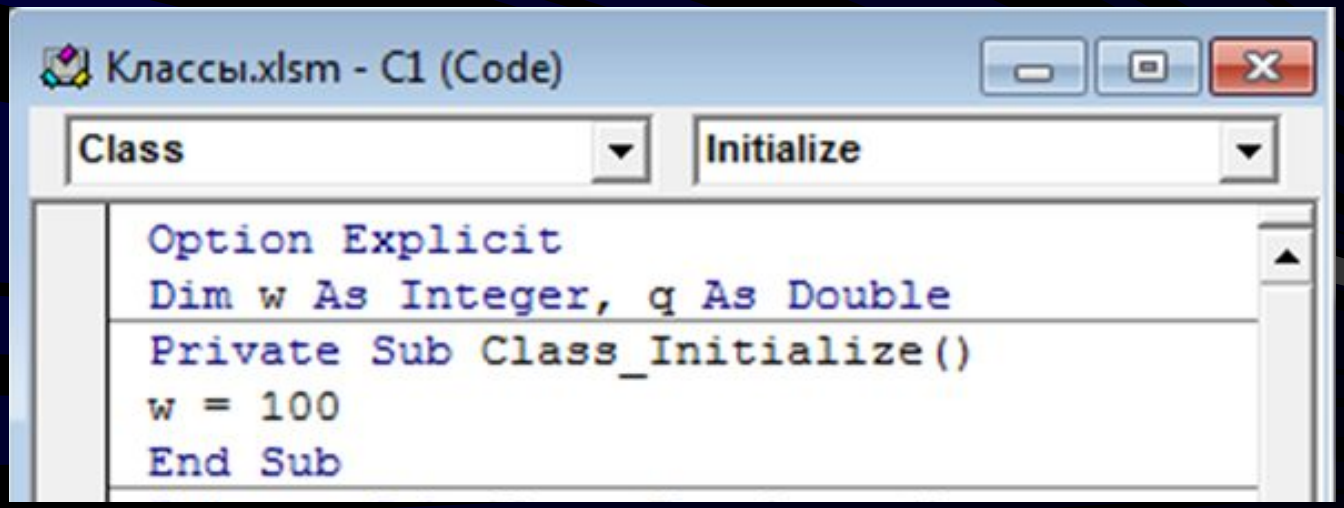
Описание переменных класса

- Многократное создание объектов приводит к многократному резервированию памяти. Поэтому не нужные объекты должны своевременно удаляться, что, как следствие, освобождает память машины.

4.2.3 Определение процедур создания и удаления объектов класса

- При создании нового объекта класса каждый раз автоматически выполнится процедура

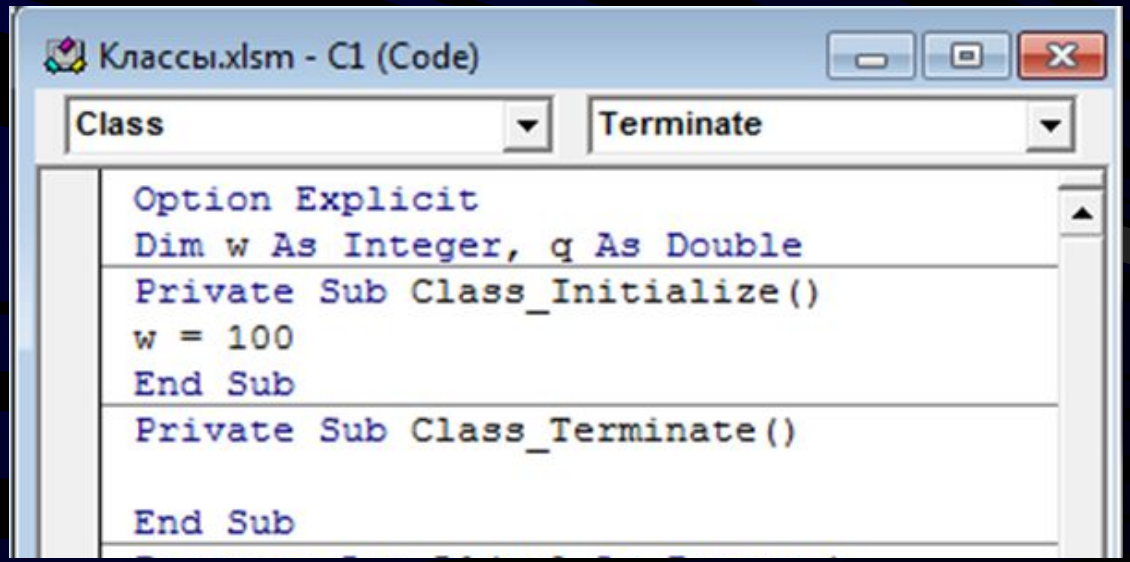
  Sub Class_Initialize()
- Программируя эту процедуру можно задать последовательность действий, которая будет выполняться с новым объектом. К числу таких действий могут относиться, например, задание размеров динамических массивов, установка начальных значений переменных и т.п.



Определение процедур создания и удаления объектов класса

- При удалении объекта класса каждый раз автоматически выполнится процедура

  Sub Class_Terminate ()
- Программируя эту процедуру можно задать последовательность действий, которая будет выполняться перед удалением объекта. К числу таких действий может быть отнесен, например, запрос о необходимости сохранения результатов работы на диске и собственно выполнение такого сохранения при положительном ответе оператора



4.2.4 Определение свойств класса

- Свойства классов задаются в виде набора функций специального вида
- Эти функции могут выполнять любые разрешенные правилами языка действия над переменными класса и, как следствие, изменять состояние объекта
- В остальном функции свойств класса не отличаются от обычных функций VBA

Определение свойств класса

Формат функции, позволяющей задавать значение переменным класса (задать значение свойства)

[**Public** | **Private**] [**Static**] **Property Let** Имя [(Список_аргументов)]
    [инструкции]
    [Exit Property]
    [инструкции]
End Property

Определение свойств класса

Формат функции, позволяющей считывать значение переменных класса (вернуть значение свойства)

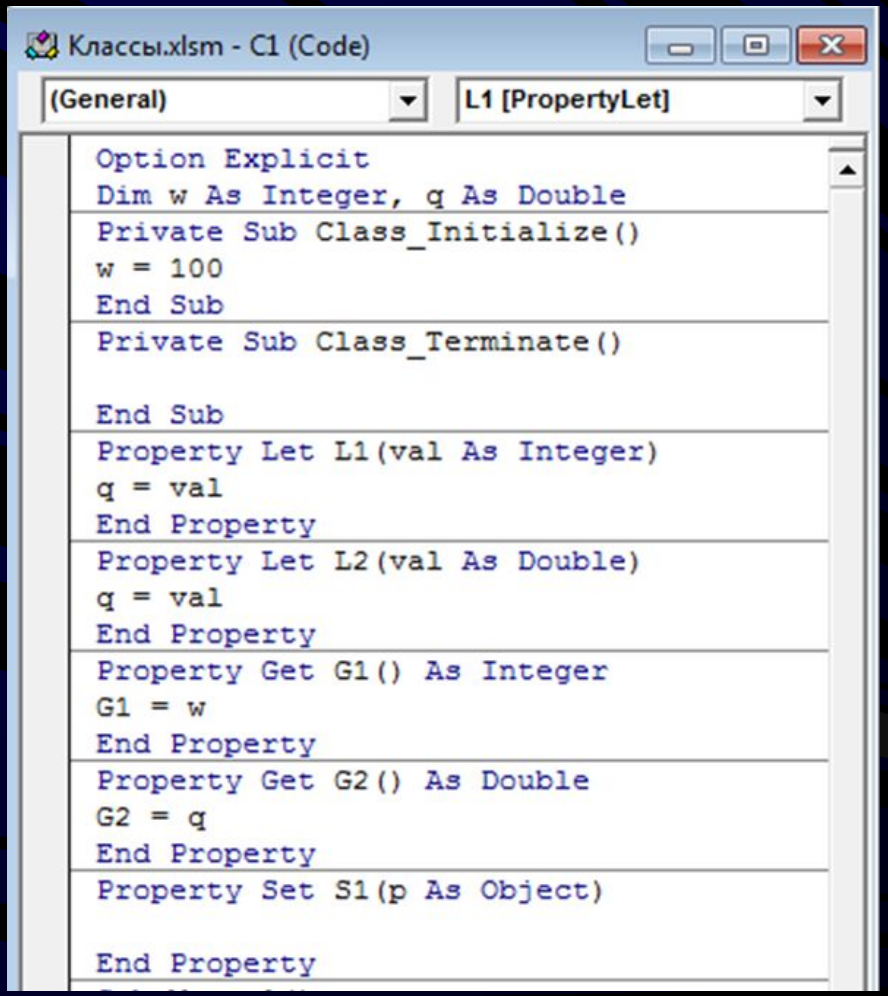
[**Public** | **Private**] [**Static**] **Property Get** Имя [(Список_аргументов)] **As** Тип
    [инструкции]
    [Exit Property]
    [инструкции]
    [Имя **=** выражение]
End Property

Определение свойств класса

Формат функции, позволяющей задать ссылку на объект

[Public | **Private**] [**Static**] **Property Set** имя [(Список_аргументов)]
    [инструкции]
    [Exit Property]
    [инструкции]
End Property

Определение свойств класса



Определение свойств класса

- Функции свойств могут иметь несколько параметров. Когда используется несколько параметров, то аргументы парных свойств должны быть согласованы

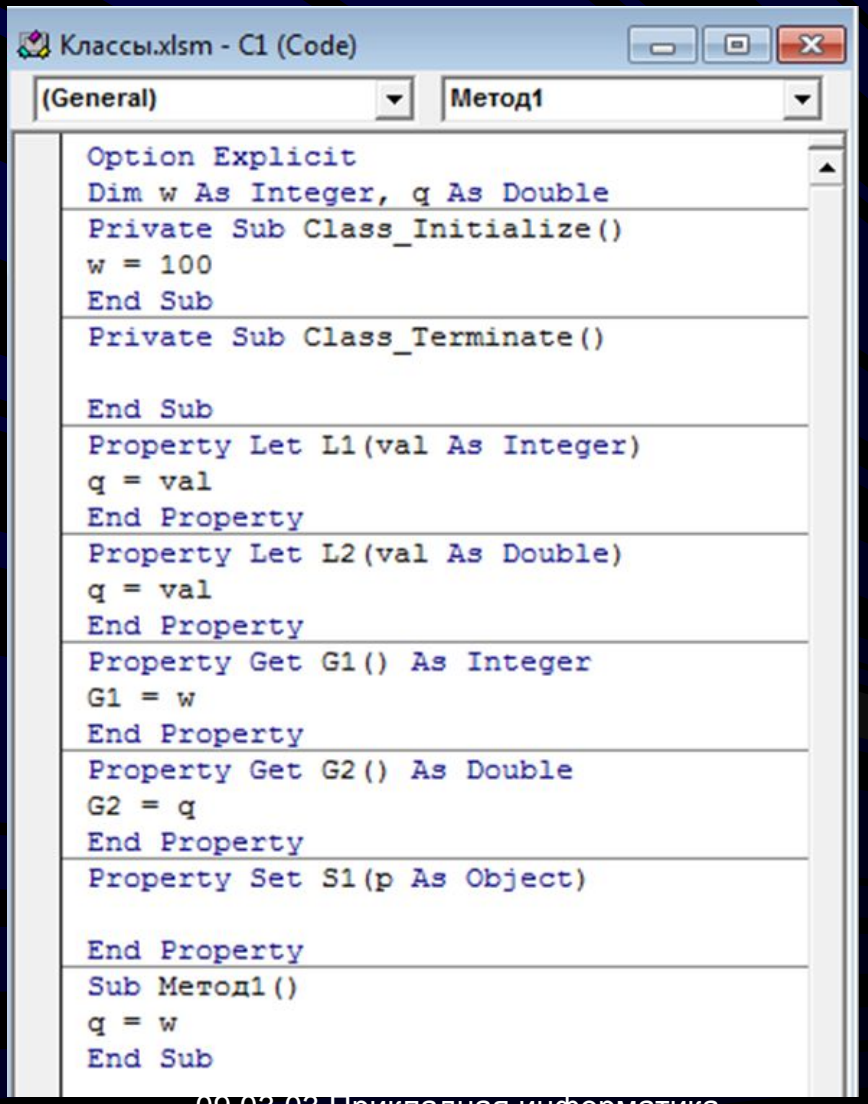
Property Let Имя **(**Аргумент**1 As** Тип**1,** Аргумент**2 As** Тип**2, ... ,** Аргумент**N As** Тип**N,** Аргумент**NN As** Тип**NN)**

Property Get Имя **(**Аргумент**1 As** Тип**1,** Аргумент**2 As** Тип**2, ... ,** Аргумент**N As** Тип**N) As** Тип**NN**

4.2.5 Определение методов класса

- Метод - действие, выполняемое над объектом.
- Метод класса задается за счет включения в текст модуля класса процедуры
- Имя метода совпадает с именем процедуры

Определение методов класса

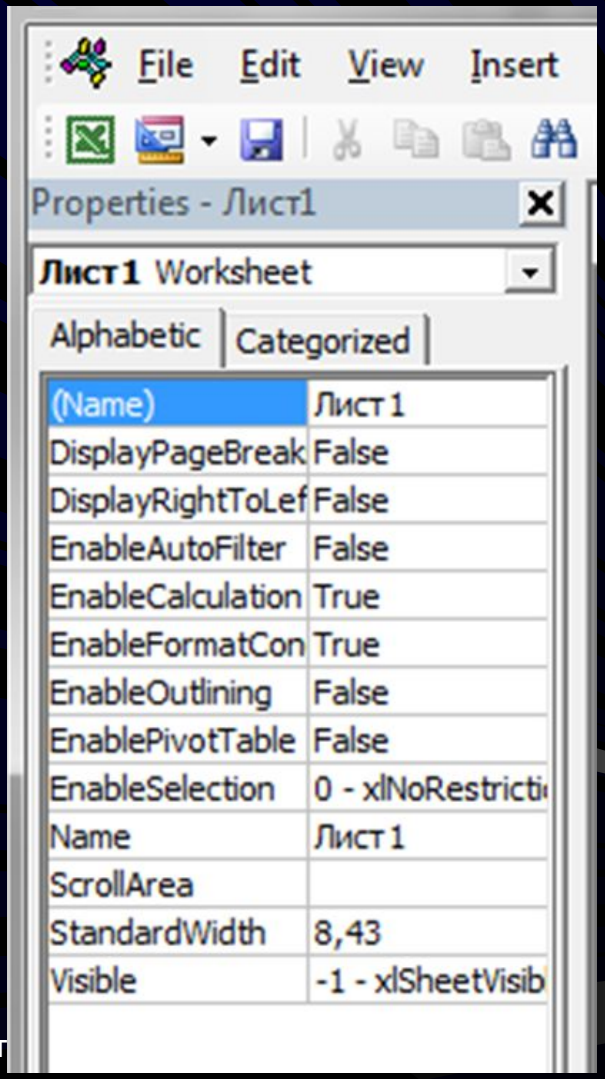


4.2.6. Библиотечные классы VBA и связанные с ними события

- При работе с VBA программисту оказывается доступно большое количество различных библиотечных классов
- Фактически подавляющее большинство возможностей, например, Excel, реализовано через классы

Библиотечные классы VBA и связанные с ними события

- Если используется библиотечный класс, то при указании на соответствующий объект оказывается активным окно его свойств



Библиотечные классы VBA и связанные с ними события

- Разработчики библиотечных классов предусмотрели возможность возникновения с объектами различного рода событий
- Событие представляет собой действие, распознаваемое объектом
- Перечень возможных событий определяется на этапе разработки

Библиотечные классы VBA и связанные с ними события

- Обработка события представляет собой программу, называемую процедурой обработки события
- Программа обработки событий может быть написана применительно к известным событиям определенных библиотечных классов

Библиотечные классы VBA и связанные с ними события

- В пользовательских классах в качестве событий могут рассматриваться события входящих в пользовательский класс объектов библиотечных классов
- Специальные средства разработки программ обработки событий пользовательских классов неизвестны

4.3. Создание объектов и работа с ними в пользовательской программе

В подразделе рассматривается:

- Объявление переменной класса в пользовательской программе
- Оператор присваивания Set
- Использование свойств класса в пользовательской программе
- Использование методов класса в пользовательской программе
- Обработка событий объекта
- Объектно-ориентированное программирование и VBA

4.3.1 Объявление переменной класса в пользовательской программе

- Если создается пользовательская программа, в составе которой планируется использовать объекты созданные пользователем или библиотечных классов, то для обращения к этим объектам в программе должны быть объявлены переменные типа используемого класса
- Допустим, что в проекте существует модуль пользовательского класса C1. Тогда объявление новой переменной имеет вид:

Dim ZZ As C1

- Тип переменной соответствует созданному пользовательскому или используемому библиотечному классу

4.3.2. Оператор присваивания Set

- Оператор Set предназначен для записи в предварительно объявленную переменную, указанную слева от символа равенства значения адреса размещения в памяти переменной, указанной справа от символа равенства

Set ZZ = Имя объекта

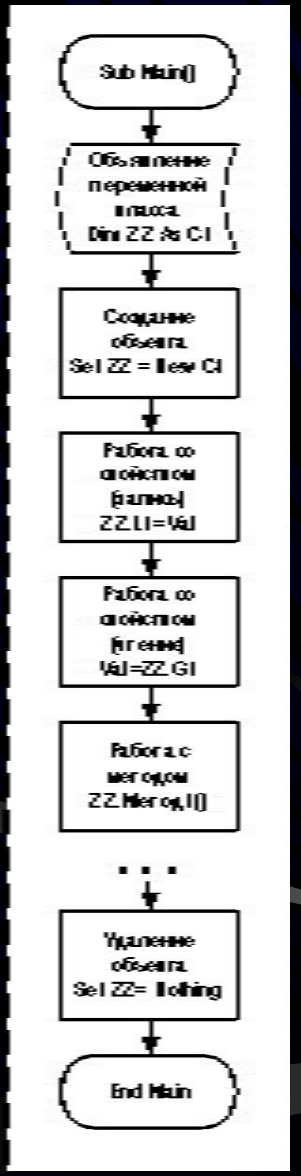
- Переменная, записанная слева от символа равенства, рассматривается как указатель. В результате выполнения оператора Set реализуется операция взятия адреса переменной, указанной справа от символа равенства, которая в свою очередь является объектом
- Поскольку объекты в VBA рассматриваются как некая совокупность данных, то для их размещения в памяти ЭВМ начиная с определенного адреса резервируется последовательный набор ячеек. Адрес первой ячейки является адресом объекта. Именно он и заносится в указатель

Оператор присваивания Set

- Ключевое слово **New** применяется в составе оператора Set для создания нового объекта а соответствии с его определением в своем классе **Set** Объект = **New** Имя_класса

Set ZZ = New C1

- В момент создания нового объекта резервируется память для его хранения. Таким способом создается новый экземпляр объекта класса C1

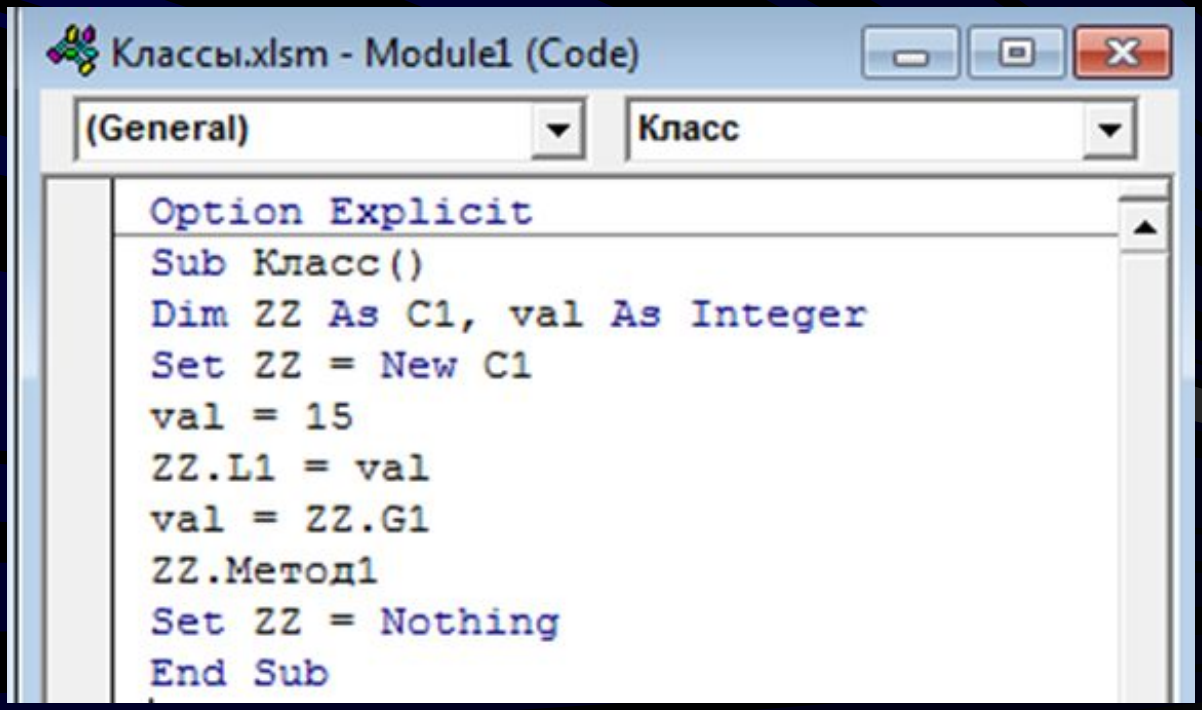


Оператор присваивания Set

- Удаление объекта

Set Объект = **Nothing**

- В результате выполнения оператора освобождаются все системные ресурсы и ресурсы памяти, выделенные для объекта



4.3.3. Использование свойств класса в пользовательской программе

- Допустим, что существует некий пользовательский класс **Панель_управления.** Создадим на его основе объект **Новая_панель.** Для этого объявим переменную Новая панель

  Dim Новая панель As Панель_управления

- Создадим объект

  Новая панель = New Панель_управления

- Ознакомившись со списком свойств и методов класса, выполняем действия с объектом. Синтаксис установки значения свойства:

  Объект.Свойство = ЗначениеСвойства

- Синтаксис чтения значения свойства:

  ЗначениеСвойства = Объект.Свойство

Использование свойств класса в пользовательской программе

- Допустим, что в классе **Панель_управления** определено свойство Цвет
- Для этого в модуле класса была создана функция

Property Let Цвет **(**Номер **As Integer)**

- Предположим, что функция **Цвет** изменяет значение переменной класса в соответствии со значением Номер, а в свою очередь ее значение используется для задания цвета панели управления при ее выводе на экран
- Тогда для задания цвета панели управления достаточно в пользовательской программе написать

Новая_Панель**.**Цвет **= 3**

Использование свойств класса в пользовательской программе

- Если ведется работа с несколькими различными свойствами одного и того же объекта, то можно воспользоваться оператором **With**

```
With Новая_Панель
    .Цвет = 3
    .Шрифт = 12
    .Кнопок = 4
End With
```

Использование свойств класса в пользовательской программе

- Пусть существует функция, возвращающая значение переменной класса

Property Get Цвет **()**

- Тогда для обращения к ней достаточно:

Текущий_цвет **=** Новая_Панель**.** Цвет

- Здесь Текущий_цвет переменная пользовательской программы

4.3.4 Использование методов класса в пользовательской программе

- Использование методов классов аналогично использованию процедур при программировании. Основное отличие заключается в том, что должен быть указан объект, к которому применяется метод.
- Предполагается, что этот объект был заранее объявлен и создан
- Синтаксис применения метода в VBA:

Объект.Метод

Пример:

Новая_Панель**. Show**

4.3.5. Обработка событий объекта

- Если в состав класса входят объекты, для которых предусмотрены некоторые события, то эти события могут быть обработаны
- Наиболее распространена обработка событий вызываемых элементами управления и представляющих собой библиотечные объекты внедренные в пользовательскую программу
- Механизма создания собственных событий применительно к произвольному объекту нет (или он не был мною найден)

4.3.6 Объектно-ориентированное программирование и VBA

Главные элементы	• Абстрагирование • Инкапсуляция • Модульность • Иерархия
Дополнительные элементы	• Типизация • Параллелизм • Сохраняемость

Объектная модель по Гради Бучу.

Объектно-ориентированное программирование и VBA

- *Абстракция* выделяет существенные характеристики некоторого объекта, отличающие его от всех других видов объектов и, таким образом, четко определяет его концептуальные границы с точки зрения наблюдателя.
- *Инкапсуляция* - это процесс отделения друг от друга элементов объекта, определяющих его устройство и поведение; инкапсуляция служит для того, чтобы изолировать контрактные обязательства абстракции от их реализации.
- *Модульность* - это свойство системы, которая была разложена на внутренне связные, но слабо связанные между собой модули.
- *Иерархия* - это упорядочение абстракций, расположение их по уровням. *Наследование* – возможность использования уже определенных объектов для построения новых объектов, т.е. возможность создания иерархии объектов. Каждый из “наследников” наследует описание данных «прародителя» и получает доступ к его свойствам и методам.

Объектно-ориентированное программирование и VBA

- *Типизация* - это способ защититься от использования объектов одного класса вместо другого, или по крайней мере управлять таким использованием
- *Параллелизм* - это свойство, отличающее активные объекты от пассивных
- *Сохраняемость* - способность объекта существовать во времени, переживая породивший его процесс, и (или) в пространстве, перемещаясь из своего первоначального адресного пространства.

Объектно-ориентированное программирование и VBA

- *Наследование* – возможность использования уже определенных объектов для построения новых объектов, т.е. возможность создания иерархии объектов. Каждый из “наследников” наследует описание данных «прародителя» и получает доступ к его свойствам и методам.
- *Полиморфизм* – возможность определения единого имени метода, применимого одновременно ко всем объектам иерархии, причем каждый из объектов иерархии может иметь свою особенность реализации этого метода. Однако Visual Basic for Applications не поддерживает механизма полиморфизма.
- *Модульность* - свойство программ, при котором объекты заключают в себе полное определение их характеристик, никакие определения методов и свойств не должны располагаться вне его, это делает возможным свободное копирование и внедрение одного объекта в другие.

Объектно-ориентированное программирование и VBA

- В VBA реализуются так называемые методы раннего и позднего связывания, причем полиморфизм обеспечивают как раз методы позднего связывания
- Раннее связывание – на этапе компиляции
- Позднее связывание – на этапе выполнения

Объектно-ориентированное программирование и VBA

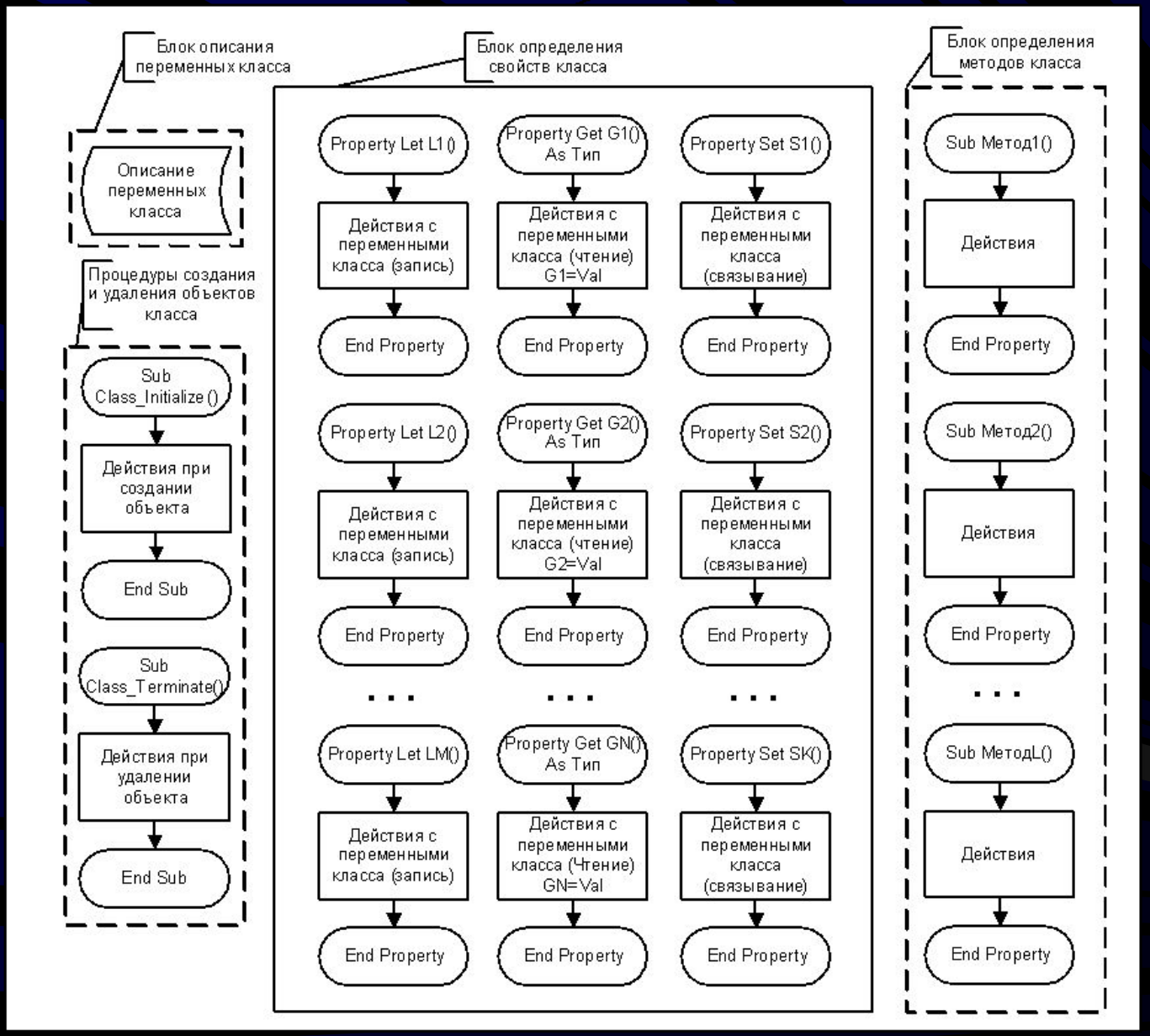
- Для реализации позднего связывания тип объекта, к которому применяется свойство или метод, объявляется как **As Object**
- Если на этапе выполнения оказывается, что конкретный объект не поддерживает вызываемого свойства, то возникает ошибка этапа выполнения

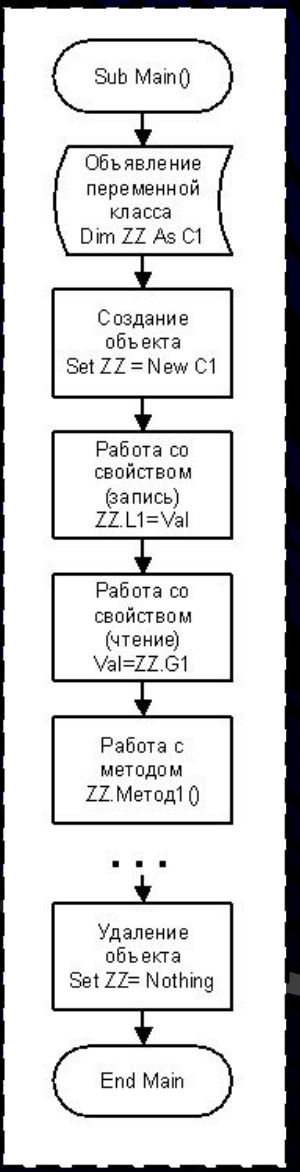
Объектно-ориентированное программирование и VBA

- Большинство объектно-ориентированных программных систем реализуют полиморфизм через наследование
- Наследование – это механизм получения нового класса из существующего

Объектно-ориентированное программирование и VBA

- В чистом виде механизма наследования в VBA нет (или мною он не найден)
- Реализация полиморфизма в VBA отлична от полиморфизма, например, C++.
- Это обстоятельство позволяет некоторым авторам утверждать, что язык VBA не является объектно-ориентированным


Блок описания переменных класса
Описание переменных класса
Процедуры создания и удаления объектов класса
Sub Class_Initialize()
Действия при создании объекта
End Sub
Sub Class_Terminate()
Действия при удалении объекта
End Sub
Блок определения свойств класса
Property Let L1()
Действия с переменными класса (запись)
End Property
Property Get G1() As Тип
Действия с переменными класса (чтение) G1=Val
End Property
Property Set S1()
Действия с переменными класса (связывание)
End Property
Property Let L2()
Действия с переменными класса (запись)
End Property
Property Get G2() As Тип
Действия с переменными класса (чтение) G2=Val
End Property
Property Set S2()
Действия с переменными класса (связывание)
End Property
Property Let LM()
Действия с переменными класса (запись)
End Property
Property Get GN() As Тип
Действия с переменными класса (чтение) GN=Val
End Property
Property Set SK()
Действия с переменными класса (связывание)
End Property
Блок определения методов класса
Sub Метод1()
Действия
End Property
Sub Метод2()
Действия
End Property
Sub МетодL()
Действия
End Property


Sub Main()
Объявление переменной класса Dim ZZ As C1
Создание объекта Set ZZ = New C1
Работа со свойством (запись) ZZ.L1=Val
Работа со свойством (чтение) Val=ZZ.G1
Работа с методом ZZ.Метод1()
Удаление объекта Set ZZ= Nothing
End Main

6. Подготовка программы к выполнению, тестирование и отладка

В разделе рассматривается

- Разработка общего алгоритма
- Стиль программирования
- Общая схема прохождения задачи
- Ошибки этапов подготовки программы к выполнению
- Ошибки этапа выполнения, автоматически определяемые процессором
- Задача тестирования
- Отладка программы

6.1. Разработка общего алгоритма

- ГОСТ 19.701-90 Единая система программной документации. Схемы алгоритмов, программ, данных и систем. Обозначения условные и правила выполнения

  В настоящем стандарте определены символы, предназначенные для использования в документации по обработке данных, и приведено руководство по условным обозначениям для применения их в:

  1) схемах данных;
  2) схемах программ;
  3) схемах работы системы;
  4) схемах взаимодействия программ;
  5) схемах ресурсов системы.

Разработка общего алгоритма

- Основные символы данных

Данные

данные, носитель данных не определен

Разработка общего алгоритма

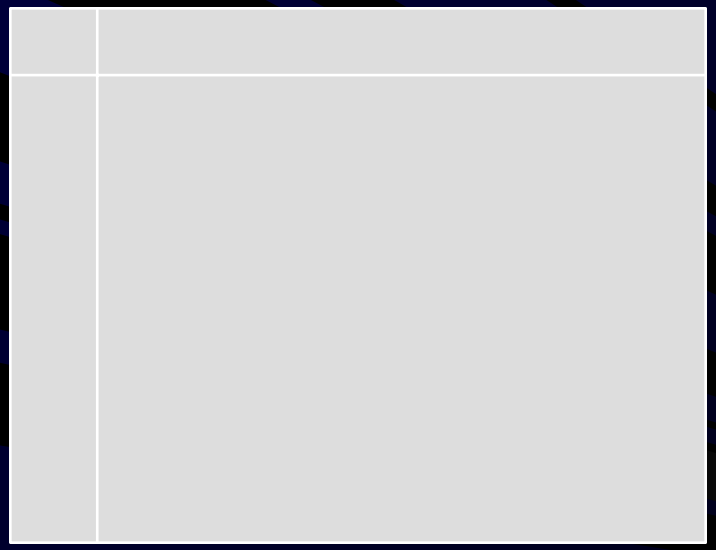
- Основные символы данных

Запоминаемые данные

Символ отображает хранимые данные в виде, пригодном для обработки, носитель данных не определен.

Разработка общего алгоритма

- Специфические символы данных

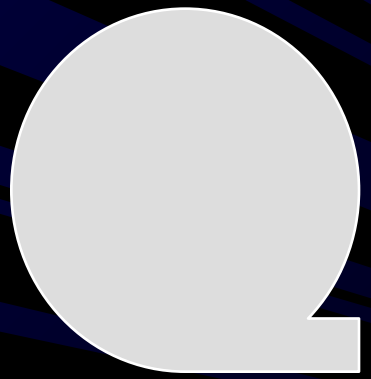


Оперативное запоминающее устройство

Символ отображает данные, хранящиеся в оперативном запоминающем устройстве

Разработка общего алгоритма

- Специфические символы данных

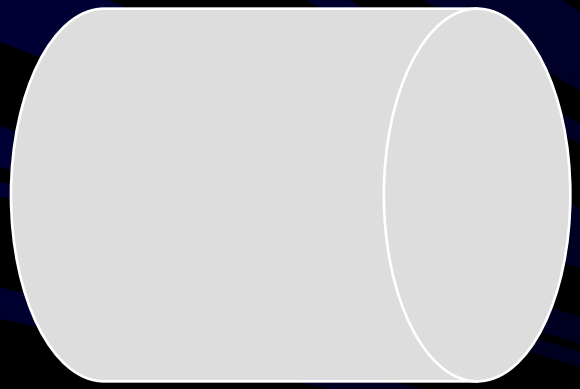


Запоминающее устройство с последовательным доступом

Символ отображает данные, хранящиеся в запоминающем устройстве с последовательным доступом
(магнитная лента, кассета с магнитной лентой, магнитофонная кассета).

Разработка общего алгоритма

- Специфические символы данных

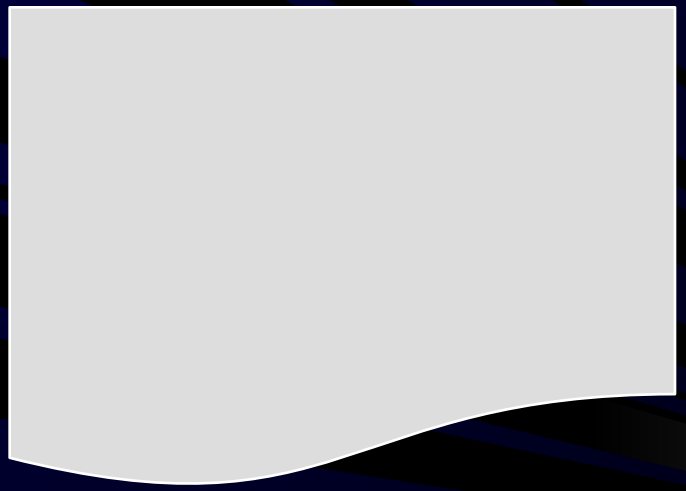


Запоминающее устройство с прямым доступом

Символ отображает данные, хранящиеся в запоминающем устройстве с прямым доступом
(магнитный диск, магнитный барабан, гибкий магнитный диск).

Разработка общего алгоритма

- Специфические символы данных



Документ

Символ отображает данные, представленные на носителе в удобочитаемой форме (машинограмма, документ для оптического или магнитного считывания, микрофильм, рулон ленты с итоговыми данными, бланки ввода данных).

Разработка общего алгоритма

- Специфические символы данных

Ручной ввод

Символ отображает данные, вводимые вручную во время обработки
с устройств любого типа
(клавиатура, переключатели, кнопки, световое перо, полоски со штриховым кодом).

Разработка общего алгоритма

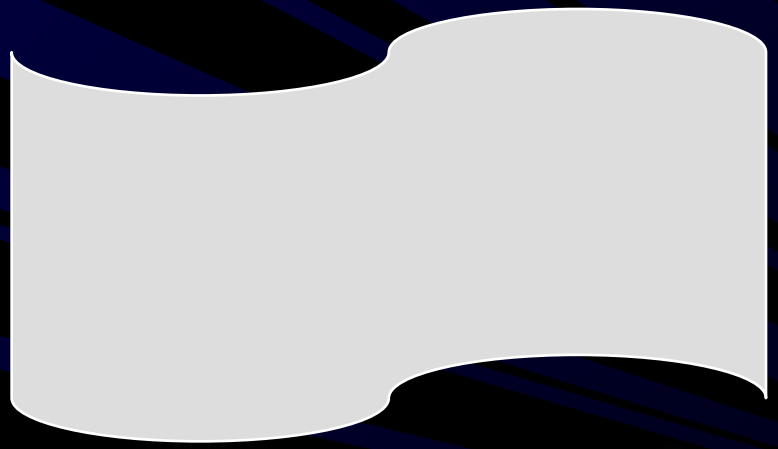
- Специфические символы данных

Карта

Символ отображает данные, представленные на носителе в виде карты (перфокарты, магнитные карты, карты со считываемыми метками, карты с отрывным ярлыком, карты со сканируемыми метками).

Разработка общего алгоритма

- Специфические символы данных



Бумажная лента

Символ отображает данные, представленные на носителе в виде бумажной ленты

Разработка общего алгоритма

- Специфические символы данных

Дисплей

Символ отображает данные, представленные в человекочитаемой форме на носителе в виде отображающего устройства (экран для визуального наблюдения, индикаторы ввода информации).

Разработка общего алгоритма

- Основные символы процесса

Процесс

Символ отображает функцию обработки данных любого вида (выполнение определенной операции или группы операций, приводящее к изменению значения, формы или размещения информации или к определению, по которому из нескольких направлений потока следует двигаться).

Разработка общего алгоритма

- Специфические символы процесса

Предопределенный процесс

Символ отображает предопределенный процесс, состоящий из одной или нескольких операций или шагов программы, которые определены в другом месте (в подпрограмме, модуле).

Разработка общего алгоритма

- Специфические символы процесса

Ручная операция

Символ отображает любой процесс, выполняемый человеком.

Разработка общего алгоритма

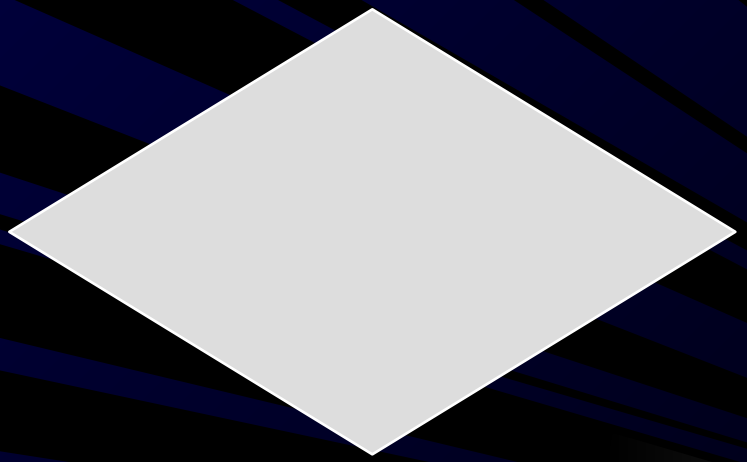
- Специфические символы процесса

Подготовка

Символ отображает модификацию команды или группы команд с целью воздействия на некоторую последующую функцию
(установка переключателя, модификация индексного регистра или инициализация программы).

Разработка общего алгоритма

- Специфические символы процесса



Решение

Символ отображает решение или функцию переключательного типа, имеющую один вход и ряд альтернативных выходов, один и только один из которых может быть активизирован после вычисления условий, определенных внутри этого символа. Соответствующие результаты вычисления могут быть записаны по соседству с линиями, отображающими эти пути.

Разработка общего алгоритма

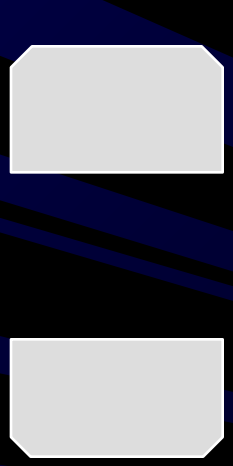
- Специфические символы процесса

Параллельные действия

Символ отображает синхронизацию двух или более параллельных операций.

Разработка общего алгоритма

- Специфические символы процесса



Граница цикла

Символ, состоящий из двух частей, отображает начало и конец, цикла. Обе части символа имеют один и тот же идентификатор. Условия для инициализации, приращения, завершения и т. д. помещаются внутри символа в начале или в конце в зависимости от расположения операции, проверяющей условие

Разработка общего алгоритма

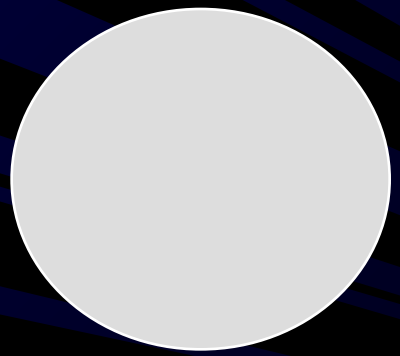
- Основной символ линий

Линия

Символ отображает поток данных или управления

Разработка общего алгоритма

- Специальные символы



Соединитель

Символ отображает выход в часть схемы и вход из другой части этой схемы и используется для обрыва линии и продолжения ее в другом месте. Соответствующие символы – соединители должны содержать одно и то же уникальное обозначение.

Разработка общего алгоритма

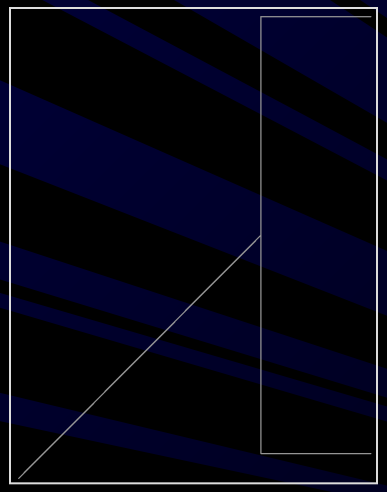
- Специальные символы

Терминатор

Символ отображает выход во внешнюю среду и вход из внешней среды (начало или конец схемы программы, внешнее использование и источник или пункт назначения данных).

Разработка общего алгоритма

- Специальные символы



Комментарий

Символ используют для добавления описательных комментариев или пояснительных записей в целях объяснения или примечаний. Пунктирные линии в символе комментария связаны с соответствующим символом или могут обводить группу символов. Текст комментариев или примечаний должен быть помещен около ограничивающей фигуры.

Разработка общего алгоритма

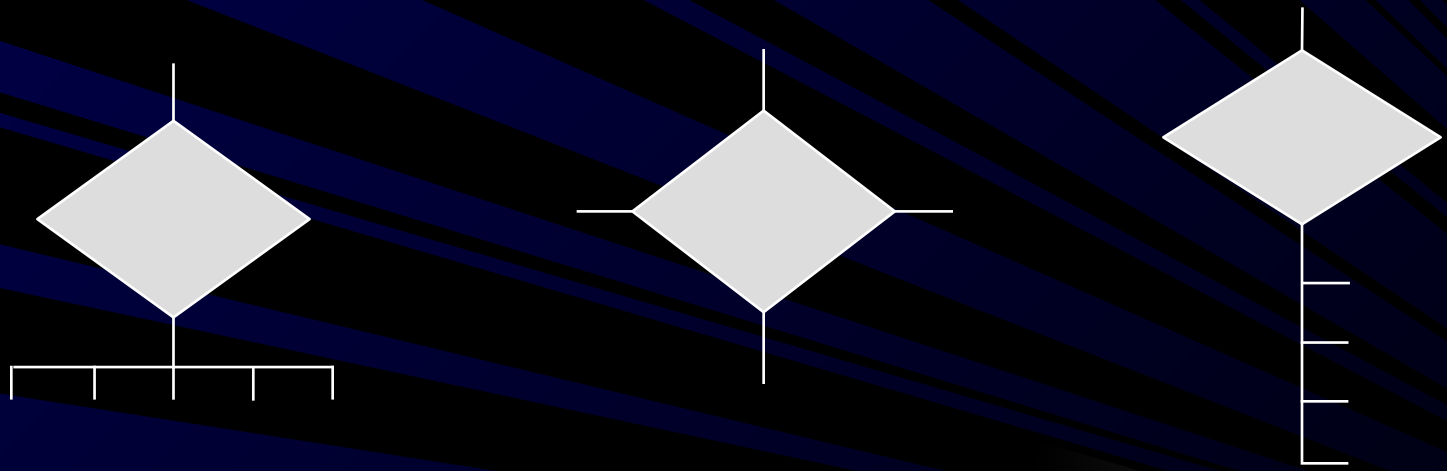
- Пропуск

Комментарий

Символ (три точки) используют в схемах
для отображения пропуска символа или группы символов,
в которых не определены ни тип, ни число символов.
Символ используют только в символах линии или между ними. Он применяется
главным образом в схемах, изображающих общие
решения с неизвестным числом повторений

Разработка общего алгоритма

- Специальные условные обозначения



Несколько выходов

Несколько выходов из символа следует показывать:
1) несколькими линиями от данного символа к другим символам;
2) одной линией от данного символа, которая затем разветвляется в соответствующее число линий

Разработка общего алгоритма

- Повторяющееся представление
  - Вместо одного символа с соответствующим текстом могут быть использованы несколько символов с перекрытием изображения, каждый из которых содержит описательный текст (использование или формирование нескольких носителей данных или файлов, производство множества копий печатных отчетов или форматов перфокарт)
  - Когда несколько символов представляют упорядоченное множество, это упорядочение должно располагаться от переднего (первого) к заднему (последнему).

Разработка общего алгоритма

- Начните с того, что попытайтесь полностью понять условие задачи. На первый взгляд это требование кажется тривиальным, однако существует достаточно много примеров ситуаций, когда программисты делали одно, а получилось совсем другое. Ну и уж совсем глупо получается, когда делали сами не понимая что.

Разработка общего алгоритма

- Еще раз обратите внимание на то, что процессор может, в сущности, выполнить только следующие действия:
  - Запись числа в определенную ячейку памяти;
  - Считывание числа из определенной ячейки памяти;
  - Выполнения некого действия со считанным на предыдущем шаге (шагах) числом (числами)

Разработка общего алгоритма

- Поэтому создаваемый вами алгоритм в пределе должен быть детализирован до уровня элементарных действий процессора

Разработка общего алгоритма

- Операции и операторы языка программирования (например, VBA) позволяют сразу принять в рассмотрение целую последовательность элементарных операций процессора
- Это означает, что для того, чтобы ими воспользоваться и в них составить алгоритм необходимо четко понимать механику их выполнения

Разработка общего алгоритма

- Определите, что является исходными данными задачи, а что есть результат ее решения
- Задумайтесь над возможным диапазоном изменения данных
- Классифицируйте типы числовых значений переменных (целые, рациональные, комплексные и т.п.)
- Подберите типы данных, требуемые для решаемой вами задачи.

Разработка общего алгоритма

- Выберите подходящий способ ввода исходных данных
- Определите способ вывода результатов работы программы

Разработка общего алгоритма

- Определите необходимую для конкретной задачи последовательность действий с исходными данными
- Попытайтесь решить задачу вручную, например, с помощью карандаша и бумаги

Разработка общего алгоритма

- Если вам непонятно, как решить задачу вручную, то необходимо разобраться с методами ее решения
- Не надейтесь, что компьютер сделает что-то за вас. Все его действия строго регламентированы и алгоритм за вас он составить не может
- Всегда добивайтесь ситуации, при которой вы в состоянии вручную получить набор выходных данных, соответствующих, по крайней мере, одному нетривиальному набору входных

Разработка общего алгоритма

- Если последовательность действий для ручного счета определена, необходимо задуматься о подборе операторов языка программирования, реализующих требуемую вам последовательность ручных действий

Разработка общего алгоритма

- Если ни один из известных операторов в вам не подходит, то имейте в виду, что набор операторов языка программирования появился вовсе не случайно. Он удовлетворяет почти всем практическим случаям, в том числе, скорее всего, и вашему.
- Поэтому вам придется еще раз задуматься о назначении каждого из операторов языка и подобрать необходимый. Может оказаться, что для этого целесообразно отложить решение задачи и еще раз перечитать соответствующий раздел руководства по программированию или учебника

Разработка общего алгоритма

- После того, как операторы и данные решаемой задачи определены, начинается этап кодирования создаваемой программы
- Только после этого вы начинаете непосредственно работать с интегрированной средой разработки (в нашем случае VBA)

6.2. Стиль программирования

- Цель программирования - не создание программы, а получение результатов вычислений
- Программы читаются людьми
- Стандартизация стиля. Если существует более одного способа сделать что-либо и выбор произвольный - остановитесь на одном способе и всегда его придерживайтесь

Стиль программирования

- Комментарии. Делайте комментариев больше, чем вам кажется необходимым.
- Комментарии должны содержать некоторую дополнительную информацию, а не перефразировать программу

Стиль программирования

- Располагайте комментарии таким образом, чтобы это не делало программу менее наглядной
- Неправильные комментарии хуже, чем их отсутствие

Стиль программирования

- Оформление текста программы
- Вводные комментарии. Назначение, способ вызова, список и назначение формальных параметров, список используемых подпрограмм
- Пропуск строк и дополнительные пробелы как средство повышения удобочитаемости программ

Стиль программирования

- Выбор имен переменных. Используйте имена с подходящей мнемоникой.
- Не используйте имеющуюся возможность записи нескольких операторов в одной строке
- Для выявления структуры программы или данных используйте отступы

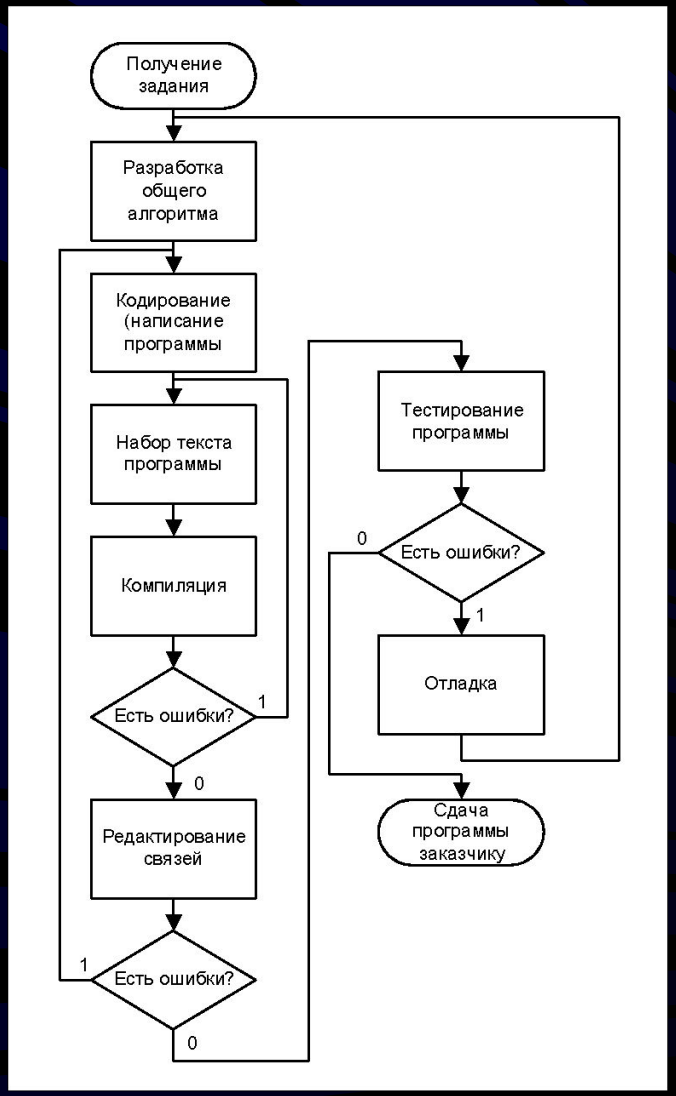
Стиль программирования

- Разбиение большой программы на разделы, подразделы и (или) подпрограммы путем выделения логических единиц улучшает восприятие программы
- Название раздела должно отражать цель данного раздела, т.е. действия, которые в нем производятся
- Программы читаются людьми

6.3. Общая схема прохождения задачи

- Общая процедура создания программного обеспечения представляет собой многошаговый процесс с большим числом обратных связей. Как правило, эта процедура выполняется за несколько шагов, причем в процессе ее выполнения приходится многократно возвращаться к ее началу.

Общая схема прохождения задачи



6.4. Ошибки этапов подготовки программы к выполнению

- В процессе преобразования текстового файла в коды, которые могут быть выполнены процессором, транслятор (компилятор, интерпретатор) может выдать разнообразные диагностические сообщения.
- В подавляющем большинстве случаев причиной появления таких сообщений является не соблюдение программистом правил языка программирования, в результате чего система не может создать последовательность исполняемых кодов.

Ошибки этапов подготовки программы к выполнению

- Если такие ошибки есть, то необходимо вернуться к исходному тексту программы и внести соответствующие изменения
- Интерпретатор VBA в этом случае непосредственно указывает строку программы, содержащую ошибку, дает ее описание и предлагает воспользоваться системой помощи.

Ошибки этапов подготовки программы к выполнению

- В некоторых случаях могут выдаваться диагностические сообщения предупреждающего или рекомендательного характера, наличие которых не останавливает процесс перевода программы в коды. Тем не менее, наличие таких сообщений является чрезвычайно серьезным сигналом и программист должен внести в текст программы изменения, которые предотвратят появление таких сообщений

6.5 Ошибки этапа выполнения, автоматически определяемые процессором

- Программа, запущенная на выполнение, может выполнить некоторые действия, которые с точки зрения разработчиков процессора являются незаконными
- Типичный пример такой операции – деление на ноль. В структуру процессора заложены проверочные действия, не допускающие возникновения подобной ситуации, и приводящие к возникновению логического прерывания процессора

Ошибки этапа выполнения, автоматически определяемые процессором

- Стандартно это прерывание обрабатывается в виде передачи управления операционной системе и, как следствие, к прекращению выполнения пользовательской программы

Ошибки этапа выполнения, автоматически определяемые процессором

- Язык VBA содержит специальный оператор обработки ошибок, позволяющий перехватывать прерывания, возникающие в процессе выполнения программы из-за ошибок этапа выполнения. Синтаксис оператора:

On Error GoTo Строка

Ошибки этапа выполнения, автоматически определяемые процессором

- Начиная с этого момента, при возникновении ошибки управление передается оператору, помеченному как Строка
- Модуль, содержащий включенный оператор обработки ошибок, должен содержать оператор **Exit** (например, **Exit Sub, Exit Function, Exit Property**), не позволяющий выполниться фрагменту программы обработки ошибок естественным путем

Ошибки этапа выполнения, автоматически определяемые процессором

- Программа обработки прерывания включается в текст модуля, начинается с первого оператора, помеченного как Строка: и заканчивается оператором **Resume Next**
- Завершение всего модуля оформляется обычным способом (например, **End Sub**)

Ошибки этапа выполнения, автоматически определяемые процессором

- Оператор **On Error Resume Next** указывает, что при возникновении ошибки управление передается на следующий оператор
- Оператор **On Error GoTo 0** отключает активизированный обработчик прерываний от ошибок в текущем модуле

Ошибки этапа выполнения, автоматически определяемые процессором

- Корректно написанная программа не должна допускать возникновение ошибок этапа выполнения за счет введения дополнительных средств внутреннего алгоритмического контроля, поэтому рассматриваемый оператор выполняет скорее отладочные, а не основные функции

6.6 Задача тестирования

- Многочисленные попытки доказать факт правильности созданной программы, к сожалению, закончились неудачей
- На настоящий момент человечество вынуждено смириться с мыслью, что программы содержат и будут содержать ошибки
- Не существует метода создания безошибочных программ и, как следствие, программирование может рассматриваться только как искусство, но не как наука

Задача тестирования

- Тестирование - процесс испытания программы на предмет ее работы с заданным набором входных данных (тестом)
- По своей сути процесс тестирования представляет собой проверку реакции программы на заранее подготовленные наборы входных данных.
- Реакция программы на тест должна быть известна до того, как она появится
- Сами тесты разрабатываются вместе с алгоритмом программы еще до начала ее кодирования

Задача тестирования

- Целью тестирования является установить факт наличия ошибки в программе
- Перед разработчиком теста стоит задача разрушить готовую программу, что само по себе является не таким уж простым делом
- Очень часто квалификация разработчика теста должна быть выше квалификации составителя программы
- Для реализации тестирования может потребоваться разработка специального программного средства (генератора тестов), которое по сложности может оказаться соизмеримым с испытуемой программой

6.7. Отладка программы

- Тест успешен, если он указал на факт наличия ошибки в программе. Это означает, что необходимо принимать меры по их устранению
- Если тест ошибок не нашел, то это означает только то, что ошибки не были найдены. К сожалению, это обстоятельство не является доказательством факта их отсутствия вообще
- Отладкой называется процесс отыскания конкретного оператора программы, являющегося причиной возникновения ошибки, и внесение в него изменений с целью устранения выявленной ошибки
- Процесс отладки завершается отысканием оператора программы, порождающего ошибку и внесением в него изменений. После этого программист снова должен вернуться к тестированию программы

6. Разработка общего алгоритма и стиль программирования

В разделе рассматривается

- Разработка общего алгоритма
- Стиль программирования

6.1. Разработка общего алгоритма

- ГОСТ 19.701-90 Единая система программной документации. Схемы алгоритмов, программ, данных и систем. Обозначения условные и правила выполнения

  В настоящем стандарте определены символы, предназначенные для использования в документации по обработке данных, и приведено руководство по условным обозначениям для применения их в:

  1) схемах данных;

  2) схемах программ;

  3) схемах работы системы;

  4) схемах взаимодействия программ;

  5) схемах ресурсов системы.

Разработка общего алгоритма

- Начните с того, что попытайтесь полностью понять условие задачи. На первый взгляд это требование кажется тривиальным, однако существует достаточно много примеров ситуаций, когда программисты делали одно, а получилось совсем другое. Ну и уж совсем глупо получается, когда делали сами не понимая что.

Разработка общего алгоритма

- Еще раз обратите внимание на то, что процессор может, в сущности, выполнить только следующие действия:
  - Запись числа в определенную ячейку памяти;
  - Считывание числа из определенной ячейки памяти;
  - Выполнения некого действия со считанным на предыдущем шаге (шагах) числом (числами)

Разработка общего алгоритма

- Поэтому создаваемый вами алгоритм в пределе должен быть детализирован до уровня элементарных действий процессора

Разработка общего алгоритма

- Операции и операторы языка программирования (например, VBA) позволяют сразу принять в рассмотрение целую последовательность элементарных операций процессора
- Это означает, что для того, чтобы ими воспользоваться и в них составить алгоритм необходимо четко понимать механику их выполнения

Разработка общего алгоритма

- Определите, что является исходными данными задачи, а что есть результат ее решения
- Задумайтесь над возможным диапазоном изменения данных
- Классифицируйте типы числовых значений переменных (целые, рациональные, комплексные и т.п.)
- Подберите типы данных, требуемые для решаемой вами задачи.

Разработка общего алгоритма

- Выберите подходящий способ ввода исходных данных
- Определите способ вывода результатов работы программы

Разработка общего алгоритма

- Определите необходимую для конкретной задачи последовательность действий с исходными данными
- Попытайтесь решить задачу вручную, например, с помощью карандаша и бумаги

Разработка общего алгоритма

- Если вам непонятно, как решить задачу вручную, то необходимо разобраться с методами ее решения
- Не надейтесь, что компьютер сделает что-то за вас. Все его действия строго регламентированы и алгоритм за вас он составить не может
- Всегда добивайтесь ситуации, при которой вы в состоянии вручную получить набор выходных данных, соответствующих, по крайней мере, одному нетривиальному набору входных

Разработка общего алгоритма

- Если последовательность действий для ручного счета определена, необходимо задуматься о подборе операторов языка программирования, реализующих требуемую вам последовательность ручных действий

Разработка общего алгоритма

- Если ни один из известных операторов в вам не подходит, то имейте в виду, что набор операторов языка программирования появился вовсе не случайно. Он удовлетворяет почти всем практическим случаям, в том числе, скорее всего, и вашему.
- Поэтому вам придется еще раз задуматься о назначении каждого из операторов языка и подобрать необходимый. Может оказаться, что для этого целесообразно отложить решение задачи и еще раз перечитать соответствующий раздел руководства по программированию или учебника

Разработка общего алгоритма

- После того, как операторы и данные решаемой задачи определены, начинается этап кодирования создаваемой программы
- Только после этого вы начинаете непосредственно работать с интегрированной средой разработки (в нашем случае VBA)

7. Проектирование программ

В разделе рассматривается:

- Общий подход к проектированию программ
- Структурная декомпозиция и структурное программирование
- Объектно-ориентированная декомпозиция

7.1. Общий подход к проектированию программ

- Программное средство – это дорогостоящая продукция и, очень часто, крупное капиталовложение

Общий подход к проектированию программ

- Стремитесь к простоте
- Проектирование программы должно начаться и закончиться до начала кодирования
- Задачи, подлежащие программированию, обычно ставятся не теми, кто будет программировать
- Когда программист и постановщик задачи не одно лицо - первый вынужден работать с представлением второго о решаемой задаче

Общий подход к проектированию программ

- Добивайтесь точности при определении задачи
- После того, как задача определена, отказывайтесь от внесения в нее изменений и дополнений. Если они все-таки необходимы, добивайтесь увеличения сметной стоимости работы и сроков на ее выполнение

Общий подход к проектированию программ

- Выбирайте алгоритм решения задачи самым тщательным образом. Час, потраченный на выбор алгоритма, стоит пяти часов программирования
- Выбирайте представление данных, соответствующее задаче
- При возможности используйте массивы и структуры, указатели и ссылки

Общий подход к проектированию программ

- Добивайтесь универсальности программы, т.е. независимости ее от конкретного набора данных
- Используйте в качестве параметров переменные, а не константы

Общий подход к проектированию программ

- Используйте существующие библиотеки программ
- Тщательно разрабатывайте форматы и вид представления входных и выходных данных. Вид выводимой информации часто является единственным критерием, по которому оценивается мастерство программиста

Общий подход к проектированию программ

- Типовой набор целей, которые устанавливаются при проектировании программы
  - высокий уровень надежности;
  - выполнение объема работ к определенной дате;
  - минимальное время разработки или минимальная стоимость;
  - удобство и простота эксплуатации;
  - эффективность;
  - возможность введения модификаций;
  - универсальность.

Общий подход к проектированию программ

- Параметры оценки программ:
  - память;
  - время;
  - сложность.
- Метод управления сложностью заключается в том, что процесс или структуру программы разбивают на небольшие, легко управляемые части, которые комбинируют для получения определенной функции

7.2. Структурная декомпозиция и структурное программирование

- В подразделе рассматривается:
- Основная задача структурного программирования
- Структурная декомпозиция и проектирование сверху вниз
- Модульное программирование
- Бригада главного программиста
- Тестирование программ

7.2.1. Основная задача структурного программирования

- Структурное программирование - метод управления сложностью в пределах каждого модуля
- Структурное программирование представляет собой метод улучшения качества программ.

Основная задача структурного программирования

- Структурное программирование сосредотачивается на одном из наиболее подверженным ошибкам факторам программ - логике программы.

Основная задача структурного программирования

- Три главных составляющих структурного программирования:
  - проектирование сверху вниз;
  - модульное программирование;
  - структурное кодирование.

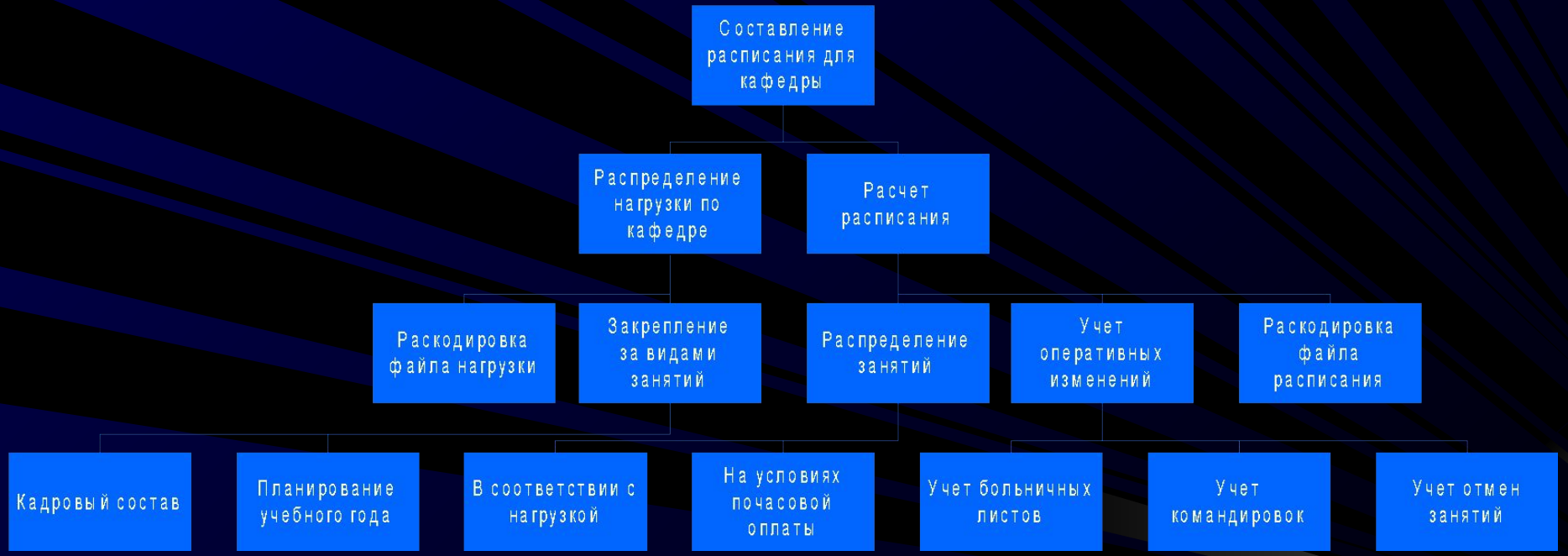
7.2.2. Структурная декомпозиция и проектирование сверху вниз

- Метод проектирования сверху вниз представляет собой сначала определение задачи в общих чертах, а затем постепенное уточнение ее структуры путем внесения более мелких деталей. Проектирование представляет собой в этом случае последовательность шагов по уточнению структуры

Структурная декомпозиция и проектирование сверху вниз

- Сначала напишите то, что вы хотите сделать, на обычном русском языке. Если вы не сможете это сделать, то вы и не сможете составить программу
- Чрезвычайно важно правильно сформулировать задачу на стадии проектирования, чтобы не исправлять ее позднее на стадиях программирования и отладки

Структурная декомпозиция и проектирование сверху вниз



- В процессе проектирования должен быть разработан интерфейс между модулями

Структурная декомпозиция и проектирование сверху вниз

- Разработка тестов должна производиться заранее
- Тестирование должно вестись параллельно с разработкой программ сверху вниз.
- Отсутствующие (не разработанные) программные модули должны заменяться заглушками

7.2.3. Модульное программирование

- Модульное программирование - процесс разделения программы на логические части, называемые модулями. При этом преследуется две цели:
  - необходимо добиться того, чтобы программный модуль не зависел от контекста, в котором он будет использоваться;
  - следует стремиться к тому, чтобы формирование программы можно было бы формировать без предварительных знаний о внутренней структуре модуля.

Модульное программирование

- Считается, что размер модуля не должен превышать 60 строк
- Модуль должен быть независим от:
  - источника входных данных;
  - места назначения выходных данных;
  - предыстории.
- Каждый модуль должен иметь свое назначение, отличающееся от назначения других модулей

Модульное программирование

- Модуль должен иметь один вход и один выход
- Связи между отдельными модулями должны быть минимизированы
- Модуль должен проверять аргументы на их принадлежность области определения. Если модуль получает значение, выходящее из области определения, то должно выдаваться сообщение об ошибке (побочный эффект)

Модульное программирование

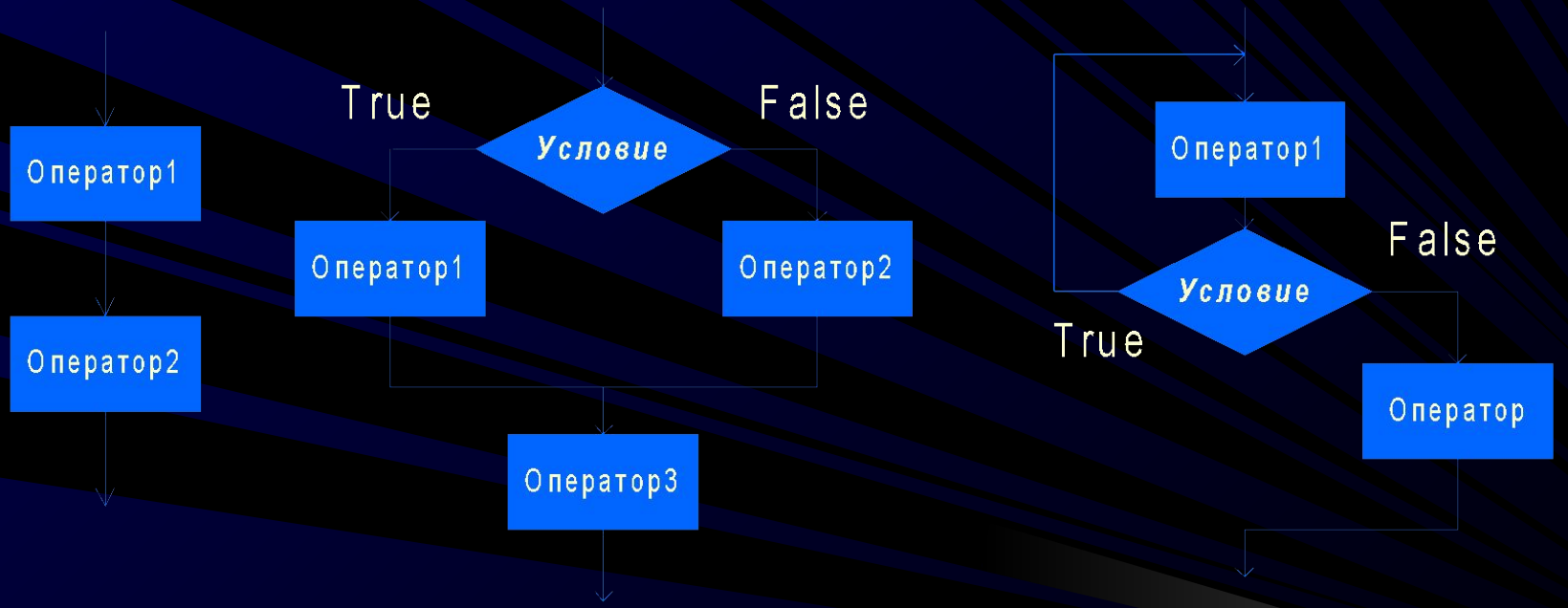
- Таким образом, для модуля должны быть определены:
  - алгоритм решения задачи;
  - область допустимых входных значений;
  - область возможных выходных значений;
  - возможные побочные эффекты.
- Возникновение побочного эффекта не должно прекращать выполнение программы

7.2.4. Структурное кодирование

- Теорема о структурировании:

  любая правильная программа с одним входом и одним выходом (без зацикливаний и недостижимых команд) может быть написана с использованием последовательности двух и более операторов, выбора одного из двух операторов, повторения выполнения оператора, пока выполняется некоторое условие

Структурное кодирование



Любая правильная программа с одним входом и одним выходом (без зацикливаний и недостижимых команд) может быть написана с использованием последовательности двух и более операторов, выбора одного из двух операторов, повторения выполнения оператора, пока выполняется некоторое условие

7.2.5. Бригада главного программиста

- Ядро бригады программистов составляют главный программист, его помощник и библиотекарь программ. Главный программист решает, когда и сколько дополнительных программистов может понадобиться. Если в бригаду потребуется много программистов, то предусматривается должность менеджера, занимающегося административными, финансовыми и правовыми вопросами

Бригада главного программиста

- Основной обязанностью главного программиста является собственно разработка и составление программ. Все члены бригады должны сообщать о ходе своей работы непосредственно ему
- Главный программист принимает все окончательные решения и отвечает за успех проекта

Бригада главного программиста

- Помощник главного программиста должен быть в состоянии осуществить руководство проектом, если это потребуется. В частности помощник главного программиста может заниматься тестирование программ, написанных главным программистом. Желательно, чтобы каждую строчку программы прочитали по крайней мере два программиста

Бригада главного программиста

- Библиотекарь хранит все записи проекта в специальной библиотеке. В библиотеке хранятся записи, показывающие текущее состояние программ и результаты их тестирования.
- Итогом работы программиста - члена бригады, является помещение его программы в библиотеку.

7.2.6. Тестирование программ

- Тестирование призвано указывать на наличие, а не на отсутствие ошибок
- Задачей программирования является не просто получение результатов, а получение *правильных* результатов
- Думать о тестировании следует на стадии написания программы задаваясь вопросом: как будет тестироваться этот сегмент?

Тестирование программ

- Необходимая полнота тестирования: должна быть испытана каждая ветвь алгоритма.
- Исчерпывающее тестирование неоправданно с экономической точки зрения и обычно неосуществимо на практике
- Проводится два вида испытаний:
  - на соответствие программы поставленной задаче;
  - на правильность ее функционирования.

Тестирование программ

- Тестирование надо начинать как можно раньше. Стоимость устранения ошибки на раннем этапе разработки существенно ниже
- Тестирование целесообразно применять по методу сверху вниз.
- Типы тестовых данных:
  - создаваемые программистом (контролируемые и случайные);
  - реальные модифицированные;
  - реальные в полном объеме.

Тестирование программ

- Каждый раз полученные в процессе тестирования данные должны быть проанализированы
- Этапы тестирования:
  - проверка в нормальных условиях;
  - проверка в экстремальных условиях;
  - проверка в исключительных условиях.

7.3. Объектно-ориентированная декомпозиция

В подразделе рассматривается:

- Причины сложности программного обеспечения
- Проектирование сложных систем
- Основные принципы построения объектно-ориентированной модели
- Объекты и классы
- Объектная декомпозиция

7.3.1. Причины сложности программного обеспечения

- Существенная черта современной программы - ее уровень сложности. Как правило, один разработчик не в состоянии охватить все аспекты системы, т.е. сложность программы превышает возможности человеческого интеллекта
- Приходится создавать новые инструменты и новую методологию проектирования программного

Причины сложности программного обеспечения

- Сложность реальной предметной области
  - Проблемы, которые люди пытаются решить с помощью программного обеспечения, часто содержат сложные элементы, а к программам предъявляется множество различных, часто взаимно исключающих требований

Причины сложности программного обеспечения

- Сложность реальной предметной области
  - Большая программная система - это крупное капиталовложение. Поэтому изменение внешних требований не должно приводить к отказу от системы в целом, в связи с чем возникает задача сопровождения программного обеспечения.

Причины сложности программного обеспечения

- Трудности управления процессом разработки
  - Основная задача разработчика - создание иллюзии простоты, в защите пользователя от сложности описываемого предмета или процесса.

Причины сложности программного обеспечения

- Трудности управления процессом разработки
  - Объем работ при разработке таков, что неизбежно требуется привлечение нескольких программистов. С целью упрощения координации их работ и уменьшения объема связей желательно, чтобы разработчиков было поменьше.

Причины сложности программного обеспечения

- Гибкость программного обеспечения.
  - Разработчик может обеспечить себя всеми необходимыми для создания системы элементами независимо от уровня абстракции. Как следствие, отсутствуют стандарты на единые конструктивные элементы и системы контроля их качества

Причины сложности программного обеспечения

- Проблема описания поведения больших дискретных систем
  - Дискретные системы, в отличие от непрерывных, имеют конечное, но чрезвычайно большое число возможных состояний

Причины сложности программного обеспечения

- Проблема описания поведения больших дискретных систем
  - Переход из одного состояния системы в другое не всегда детерминирован

Причины сложности программного обеспечения

- Проблема описания поведения больших дискретных систем
  - При неблагоприятных условиях внешнее событие может нарушить текущее состояние системы из-за того, что не были предусмотрены все возможные варианты

7.3.2. Проектирование сложных систем

- Целью проектирования системы является создание системы
  - удовлетворяющей заданным (в том числе и неформальным) требованиям (спецификациям);

Проектирование сложных систем

- Целью проектирования системы является создание системы
  - согласованной с ограничениями, накладываемыми оборудованием;
  - удовлетворяющей явным и неявным требованиям по эксплутационным качествам и ресурсопотреблению;

Проектирование сложных систем

- Целью проектирования системы является создание системы
  - удовлетворяющей явным и неявным критериям дизайна продукта;
  - удовлетворяющей требованиям к самому процессу разработки (продолжительность, стоимость и т.п.)

Проектирование сложных систем

- В основе проектирования стоит построение модели. Моделью называется некоторая абстракция, которая в той или иной степени соответствует поведению реального объекта

Проектирование сложных систем

- В основе объектно-ориентированной технологии создания программного обеспечения лежит так называемая объектная модель

7.3.3. Основные принципы построения объектно-ориентированной модели

В пункте рассматривается:

- Абстрагирование
- Инкапсуляция
- Модульность
- Иерархия
- Типизация
- Параллелизм
- Сохраняемость
- Полиморфизм

7.3.3.1. Абстрагирование

- Абстракция выделяет существенные характеристики некоторого объекта, отличающие его от всех других видов объектов и, таким образом, четко определяет его концептуальные границы с точки зрения наблюдателя.
- Суть абстракции - отделение существенных особенностей поведения от несущественных.

Абстрагирование

- Абстракция сущности - объект представляет собой полезную модель некой сущности в предметной области
- Клиентом называется любой объект, использующий ресурсы другого объекта, называемого сервером

Абстрагирование

- Поведение объекта характеризуется услугами, которые он оказывает другим объектам, и операциями, которые он выполняет над другими объектами

Абстрагирование

- Внешнее поведение объекта рассматривается с точки зрения его контракта с другими объектами. Каждая операция, предусмотренная контрактом, однозначно определена.

Абстрагирование

- Полный набор операций, которые объект может осуществлять над другим объектом, и правильный порядок их вызова называется протоколом.

Абстрагирование

- Инвариантом называется некоторое логическое условие (истина или ложь), значение которого должно сохраняться
- Для каждой операции можно задать предусловия (инварианты, предполагаемые операцией), и постусловия (инварианты, которым удовлетворяет операция

Абстрагирование

- Если нарушено предусловие - виноват клиент, постусловие - виноват сервер

Абстрагирование

- Абстракции могут обладать статическими и динамическими свойствами.
- Абстракции могут не зависеть от своего предыдущего состояния (автомат без памяти) и зависеть (автомат с памятью)

7.3.3.2. Инкапсуляция

- Инкапсуляция - процесс отделения друг от друга элементов объекта, определяющих его устройство и поведение
- Инкапсуляция служит для того, чтобы изолировать контрактные обязательства абстракции от их реализации

Инкапсуляция

- Абстрагирование направлено на наблюдаемое поведение объекта, а инкапсуляция занимается его внутренним устройством

7.3.3.3. Модульность

- Модульность - свойство системы, которая разложена на сильно связанные внутри, но слабо связанные между собой модули.
- Модуль - это самостоятельная конструкция языка.
- Интерфейс модуля отделен от его реализации, в связи с чем модуль решает задачу инкапсуляции.

7.3.3.4. Иерархия

- Иерархия - упорядочение абстракций, расположение их по уровням.
- Основными видами иерархических структур применительно к сложным объектам является структура классов и структура объектов.

Иерархия

- Под наследованием обычно понимается создание производного класса на базе другого. Базовый класс – это любой класс, который используется в качестве основы для определения другого класса. Производный класс автоматически получает данные базового класса, а также доступ к функциям-членам этого класса.

Иерархия

- Для того, чтобы указать, какой класс является производным, а какой базовым, в заголовок определения производного класса включается строка вида:
- class A: public Base {}; // Производный класс A наследующий Base

Иерархия

- Наследующий класс может использовать данные и функции базового класса и дополнять их собственными данными и функциями. Объект базового класса отличается от объекта производного класса за счет добавления в последний данных, созданных в производном классе. Это позволяет последовательно наращивать (уточнять) структуру классов и, как следствие, создавать итоговый класс сверху вниз.

Иерархия

- Наследование порождает иерархию «обобщение - специализация», в которой подкласс представляет собой частный случай своего суперкласса

7.3.3.4. Типизация

- Тип - точная характеристика свойств, включая структуру и поведение, относящуюся к некоторой совокупности объектов.
- Можно считать, что понятия типа и класса эквивалентны.

Типизация

- Типизация - способ защититься от использования объектов одного класса вместо другого, или по крайней мере управлять таким использованием.

7.3.3.5. Параллелизм

- Параллелизм - свойство, позволяющее отличать активные объекты от пассивных
- Процесс - фундаментальная единица действия в системе.
- Каждый объект может представлять собой самостоятельный процесс и может быть активным (выполнять действия) и пассивным (состояние ожидания).

Параллелизм

- Главным вопросом параллелизма является вопрос синхронизации процессов.
- Реальная параллельность может быть достигнута только на многопроцессорных системах, система с одним процессором имитирует параллельность за счет разделения времени.

7.3.3.6. Сохраняемость

- Сохраняемость - способность объекта существовать во времени, переживая породивший его процесс, и (или) в пространстве, перемещаясь из своего первоначального адресного пространства.

Сохраняемость

- Возможные варианты сохраняемости объектов:
  - промежуточные результаты вычислений;
  - локальные переменные при вызове процедур;
  - собственные переменные, глобальные переменные, динамически создаваемые данные;

Сохраняемость

- Возможные варианты сохраняемости объектов:
  - данные, сохраняемые между сеансами выполнения программы;
  - данные, сохраняемые при переходе на новую версию программы;
  - данные, которые вообще переживут программу

7.3.3.7. Полиморфизм

- Слово полиморфизм имеет греческое происхождение и может быть переведено на русский язык как многоформенность. В программировании под полиморфизмом понимают возможность объектов с одинаковой спецификацией иметь различную реализацию (форму) в процессе выполнения программы.

Полиморфизм

- Полиморфизм в C++ реализуется за счет существующей возможности создавать так называемые виртуальные (virtual) функции
- В отличие от обычных функций, коды которых формируются компилятором и размещаются в памяти редактором связей, виртуальная функция является динамической, то есть она размещается в памяти на этапе выполнения программы.

Полиморфизм

- Достоинством полиморфизма является то обстоятельство, что при использовании объекта можно вызывать определенное свойство не заботясь о том, как объект выполняет задачу

7.3.4. Объекты и классы

- С точки зрения восприятия человеком объектом может быть:
  - осязаемый и (или) видимый предмет;
  - нечто, воспринимаемое мышлением;
  - нечто, на что направлена мысль или действие.
- Объект обладает состоянием, поведением и идентичностью.
- Структура и поведение схожих объектов определяет общий для них класс.

Объекты и классы

- Класс - это некоторое множество объектов, имеющих общую структуру и поведение
- Любой конкретный объект является просто экземпляром класса
- Термины «экземпляр класса» и «объект взаимозаменяемы

Объекты и классы

- Состояние объекта характеризуется перечнем (обычно статическим) всех свойств данного объекта и текущими (обычно динамическими) значениями каждого из этих свойств.
- Поведение - это то, как объект действует и реагирует.
- Поведение выражается в терминах состояния объекта и передачи сообщений.

Объекты и классы

- Идентичность - это такое свойство объекта, которое отличает его от всех других объектов.

Объекты и классы

- Операции - это услуги, которые объект может предоставить клиентам. К их числу относятся:
  - модификатор (изменение состояния объекта);
  - селектор (считывание состояния объекта);
  - итератор (последовательный доступ к частям объекта);
  - конструктор (создание объекта и (или) его инициализация;
  - деструктор (освобождение состояния объекта и (или) его разрушение.

Объекты и классы

- Типы отношений между объектами:
  - связи;
  - агрегация.
- Связь - это физическое или концептуальное соединение между объектами, через которое клиент запрашивает услугу у объекта - сервера.

Объекты и классы

- Участвуя в связи, объект может выполнять одну из трех ролей:
  - актер (может воздействовать на другие объекты, но сам никогда не подвергается воздействию);
  - сервер (может подвергаться воздействию других объектов, но никогда не выступает в роли воздействующего объекта);
  - агент (может выступать как в активной, так и в пассивной роли).

Объекты и классы

- Агрегация объектов описывает отношения целого и части, приводящего к соответствующей иерархии. Идя от целого (агрегата) мы можем придти к его частям (атрибутам).

7.3.5. Объектная декомпозиция

- Объектно-ориентированный подход к разработке программного обеспечения предлагает способ декомпозиции сложной системы, отличный от рассмотренного нами ранее способа алгоритмической декомпозиции

Объектная декомпозиция

- Проектирование программной системы методом объектной декомпозиции заключается в построении иерархии объектов, обменивающихся между собой сообщениями.

Объектная декомпозиция

- В отличие модулей, объект обладает рядом дополнительных качеств, существенно отличающих его от модуля.
- Как и модуль, объект обеспечивает некоторую строго определенную реакцию на входное воздействие, называемую поведением объекта.

Объектная декомпозиция

- В отличие от модуля, эта реакция существенно зависит от ранее установленных параметров объекта, которые описывают его состояние. Поэтому реакция объекта на входное воздействие оказывается разной в зависимости от того, что раньше происходило с объектом.

Объектная декомпозиция

- Можно сказать, что, в отличие от модуля, являющегося по своей сути автоматом без памяти, объект может рассматриваться как некий автомат с памятью.
- В рамках такого подхода объект может быть представлен и как некая, в том числе математическая, модель реально существующего объекта.

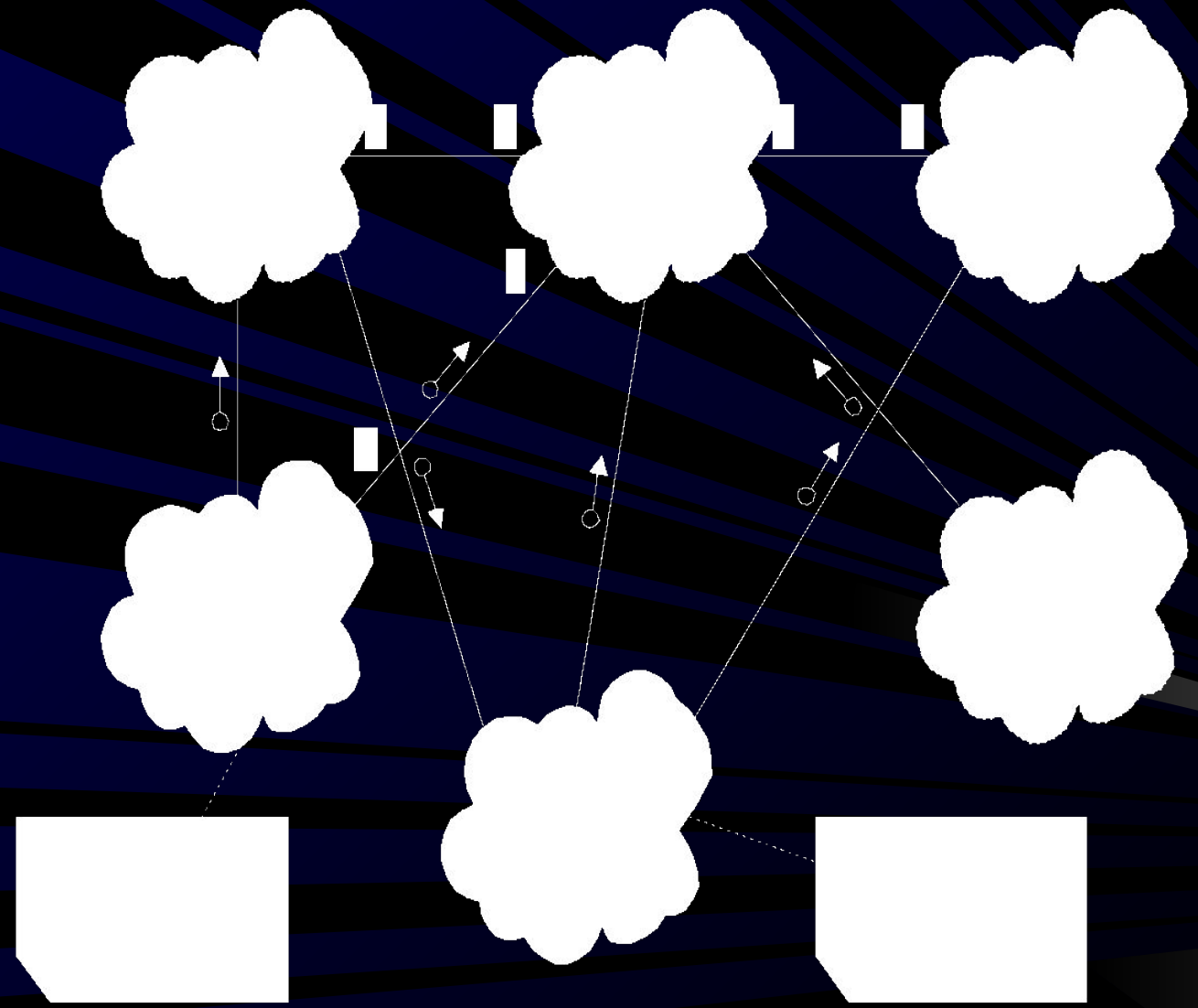
Объектная декомпозиция

- Наличие у объекта набора свойств и состояний позволяет с его помощью описывать гораздо более крупные функциональные составляющие разрабатываемой программы. С помощью объектов удобно создавать функциональные модели поведения элементов системы и, как следствие, приходить к более естественной записи ее структуры.

Объектная декомпозиция

- Результат объектной декомпозиции может получиться более простым, чем результат алгоритмической декомпозиции, хотя надо иметь в виду, что и в том, и в ином случае они описывают одно и то же.

Объектная декомпозиция



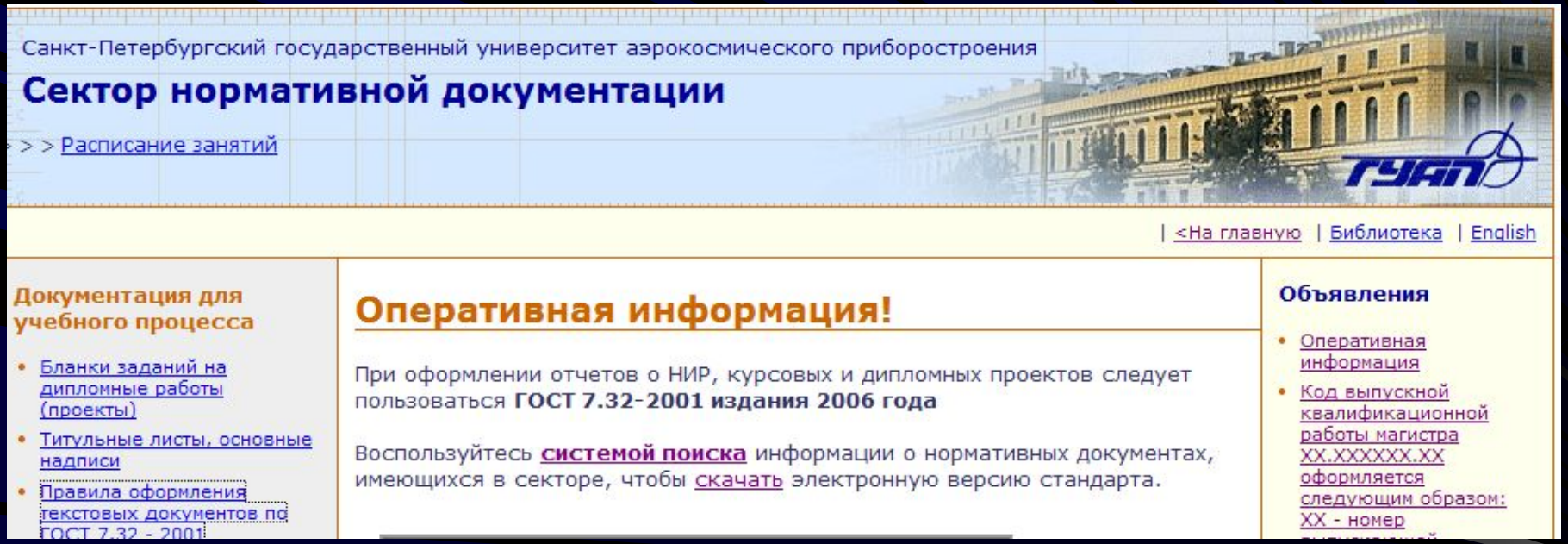
6. Создание документов средствами текстового процессора

- В разделе рассматривается:
- Нормативная документация
- Создание структуры документа
- Установка параметров страницы
- Разработка и создание колонтитулов документа
- Вставка сносок
- Создание формульных выражений
- Разработка и создание таблиц
- Разработка и создание рисунков
- Вставка ссылок на литературу
- Использование закладок
- Составление оглавления, списка таблиц и иллюстраций

Нормативная документация

- Основной задачей изучения Word является освоение методов работы с текстовыми документами большого объема

Нормативная документация



http://guap.ru/guap/standart/ob1_main.shtml

Нормативная документация

I

МЕЖГОСУДАРСТВЕННЫЙ СТАНДАРТ

**Система стандартов по информации,
библиотечному и издательскому делу**

ОТЧЕТ О НАУЧНО-ИССЛЕДОВАТЕЛЬСКОЙ РАБОТЕ

Структура и правила оформления

Издание официальное

**МЕЖГОСУДАРСТВЕННЫЙ СОВЕТ
ПО СТАНДАРТИЗАЦИИ, МЕТРОЛОГИИ И СЕРТИФИКАЦИИ
Минск**

II

Предисловие

1 РАЗРАБОТАН Всероссийским институтом научной и технической информации Российской Академии наук, Всероссийским научно-техническим информационным центром и Межгосударственным техническим комитетом по стандартизации МТК 191 «Научно-техническая информация, библиотечное и издательское дело»

ВНЕСЕН Госстандартом России

2 ПРИНЯТ Межгосударственным Советом по стандартизации, метрологии и сертификации (отчет Технического секретариата № 19 от 22 мая 2001 г.)

За принятие проголосовали

Наименование государства	Наименование национального органа по стандартизации
Азербайджанская Республика	Азгосстандарт
Республика Армения	Армгосстандарт
Республика Беларусь	Госстандарт Республики Беларусь

Нормативная документация

3 Общие положения

3.1 Отчет о НИР — научно-технический документ, который содержит систематизированные данные о научно-исследовательской работе, описывает состояние научно-технической проблемы, процесс и/или результаты научного исследования.

3.2 По результатам выполнения НИР составляется заключительный отчет о работе в целом. Кроме того, по отдельным этапам НИР могут быть составлены промежуточные отчеты, что отражается в Техническом задании на НИР и в календарном плане выполнения НИР.

3.3 Ответственность за достоверность данных, содержащихся в отчете, и за соответствие его требованиям настоящего стандарта несет организация-исполнитель.

3.4 Отчет о НИР подлежит обязательному нормоконтролю в организации-исполнителе. При проведении нормоконтроля рекомендуется руководствоваться ГОСТ 2.111.

Нормативная документация

4 Структурные элементы отчета

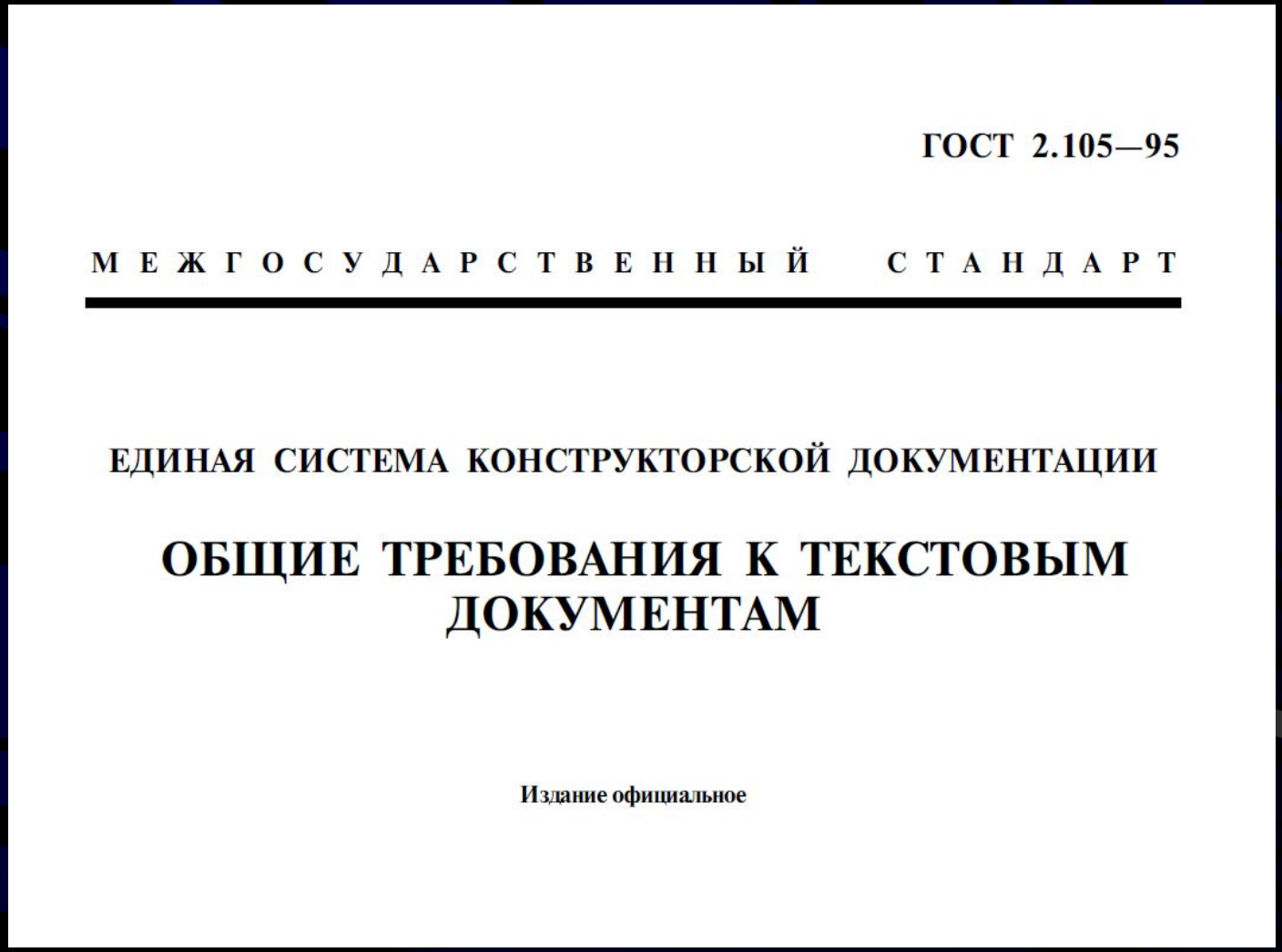
Структурными элементами отчета о НИР являются:

- **титульный лист;**
- **список исполнителей;**
- **реферат;**
- содержание;
- определения;
- обозначения и сокращения;
- **введение;**
- **основная часть;**
- **заключение;**
- список использованных источников;
- приложения.

Обязательные структурные элементы выделены полужирным шрифтом. Остальные структурные элементы включают в отчет по усмотрению исполнителя НИР с учетом требований разделов 5 и 6.

(Измененная редакция, Изм. № 1, Поправка).

Нормативная документация



ГОСТ 2.105—95

МЕЖГОСУДАРСТВЕННЫЙ СТАНДАРТ

ЕДИНАЯ СИСТЕМА КОНСТРУКТОРСКОЙ ДОКУМЕНТАЦИИ

ОБЩИЕ ТРЕБОВАНИЯ К ТЕКСТОВЫМ ДОКУМЕНТАМ

Издание официальное

Этот стандарт тоже никто не отменял.

Нормативная документация

ГОСТ 2.105—95

3 ОБЩИЕ ПОЛОЖЕНИЯ

3.1 Текстовые документы подразделяют на документы, содержащие, в основном, сплошной текст (технические условия, паспорта, расчеты, пояснительные записки, инструкции и т. п.), и документы, содержащие текст, разбитый на графы (спецификации, ведомости, таблицы и т. п.).

3.2 Текстовые документы выполняют на формах, установленных соответствующими стандартами Единой системы конструкторской документации (ЕСКД) и Системы проектной документации для строительства (СПДС).

Требования, специфические для некоторых видов текстовых документов (например эксплуатационных документов), приведены в соответствующих стандартах.

3.3 Подлинники текстовых документов выполняют одним из следующих способов:

- машинописным, при этом следует выполнять требования ГОСТ 13.1.002. Шрифт пишущей машинки должен быть четким, высотой не менее 2,5 мм, лента только черного цвета (полужирная);
- рукописным — чертежным шрифтом по ГОСТ 2.304 с высотой букв и цифр не менее 2,5 мм. Цифры и буквы необходимо писать четко черной тушью;
- с применением печатающих и графических устройств вывода ЭВМ (ГОСТ 2.004).
- на магнитных носителях данных (ГОСТ 28388).

Этот стандарт тоже никто не отменял.

Документация для учебного процесса

- Бланки заданий на дипломные работы (проекты)
- Титульные листы, основные надписи
- Правила оформления текстовых документов по ГОСТ 7.32 - 2001

Документация для научной работы

- Бланки титульных листов для отчетов о НИР
- Правила оформления отчетов о НИР
- ГОСТ 7.32-2001 Отчет о научно-исследовательской работе. Структура и правила оформления
- Примеры библиографического описания (по ГОСТ 7.1-2003)

Электронная библиотека сектора

- Общий перечень стандартов *(на 25 июня 2012 года,* 3216 наименований pdf, 834 Kb)
- Терминологические словари-справочники

Правила оформления текстовых документов

ПРАВИЛА ОФОРМЛЕНИЯ
текстовых документов по ГОСТ 7.32 - 2001

Изложение текста и оформление работ следует выполнять в соответствии с требованиями ГОСТ 7.32 - 2001

1 Текст работ следует печатать, соблюдая следующие требования:

- текст набирается шрифтом Times New Roman кеглем не менее 12, строчным, без выделения, с выравниванием по ширине;
- абзацный отступ должен быть одинаковым и равен по всему тексту 1,27 см;
- строки разделяются полуторным интервалом;
- поля страницы: верхнее и нижнее не менее 20 мм, левое не менее 30 мм, правое не менее 10 мм;
- полужирный шрифт не применяется;
- разрешается использовать компьютерные возможности акцентирования внимания на определенных терминах, формулах, теоремах, применяя шрифты разной гарнитуры;
- введение и заключение не нумеруются.

2 Основную часть работы следует делить на разделы и подразделы:

- разделы и подразделы должны иметь порядковую нумерацию в пределах всего текста, за исключением приложений;
- нумеровать их следует арабскими цифрами;
- номер подраздела включает номер раздела и порядковый номер подраздела, разделенные точкой;
- после номера раздела и подраздела в тексте точку не ставят;
- разделы и подразделы должны иметь заголовки;
- заголовки разделов и подразделов следует печатать с абзацного отступа с прописной буквы без точки в конце, не подчеркивая;
- если заголовок состоит из двух предложений, их разделяют точкой;
- переносы слов в заголовках не допускаются;

3 Нумерация страниц текстовых документов:

- страницы работ следует нумеровать арабскими цифрами, соблюдая сквозную нумерацию по всему тексту работ;
- титульный лист включают в общую нумерацию страниц работ;
- номер страницы на титульном листе не проставляют;
- номер страницы проставляют в ц[illegible] листа без точки.

@ sherstyki@mail.ru

Скачать правила оформления текстовых документов по ГОСТ 7.32 - [illegible]

Объявления

- Оперативная информация
- Код выпускной квалификационной работы магистра XX.XXXXXX.XX оформляется следующим образом: XX - номер выпускающей кафедры; XXXXXX - индекс направления, XX - номер приказа ректора ГУАП об утверждении темы
- При оформлении отчетов о НИР, курсовых и дипломных проектов следует пользоваться ГОСТ 7.32-2001 издания 2006 года
- Примеры оформления списка использованных источников по ГОСТ 7.1-2003
- *Отчет о НИР* Титульные листы

Информационные ресурсы

- Система поиска информации о нормативных документах, имеющихся в секторе (электронные и бумажные версии)

6.1. Создание структуры документа

ГОСТ 7.32 - 2001

6 Правила оформления отчета

6.1 Общие требования

6.1.1 Изложение текста и оформление отчета выполняют в соответствии с требованиями настоящего стандарта. Страницы текста отчета о НИР и включенные в отчет иллюстрации и таблицы должны соответствовать формату А4 по ГОСТ 9327. Допускается применение формата А3 при наличии большого количества таблиц и иллюстраций данного формата.

6.1.2 Отчет о НИР должен быть выполнен любым печатным способом на пишущей машинке или с использованием компьютера и принтера на одной стороне листа белой бумаги формата А4 через полтора интервала. Цвет шрифта должен быть черным, высота букв, цифр и других знаков — не менее 1,8 мм (кегль не менее 12). Полужирный шрифт не применяется.

Текст отчета следует печатать, соблюдая следующие размеры полей: правое — не менее 10 мм, верхнее и нижнее — не менее 20 мм, левое — не менее 30 мм.

Разрешается использовать компьютерные возможности акцентирования внимания на определенных терминах, формулах, теоремах, применяя шрифты разной гарнитуры.

6.1.1, 6.1.2 **(Измененная редакция, Изм. № 1).**

6.1.3 Вне зависимости от способа выполнения отчета качество напечатанного текста и оформления иллюстраций, таблиц, распечаток с ПЭВМ должно удовлетворять требованию их четкого воспроизведения.

6.1.4 При выполнении отчета необходимо соблюдать равномерную плотность, контрастность и четкость изображения по всему отчету. В отчете должны быть четкие, нерасплывшиеся линии, буквы, цифры и знаки.

6.1.5 Опечатки, описки и графические неточности, обнаруженные в процессе подготовки отчета, допускается исправлять подчисткой или закрашиванием белой краской и нанесением на том же месте исправленного текста (графики) машинописным способом или черными чернилами, пастой или тушью — рукописным способом.

Повреждения листов отчета, помарки и следы неполностью удаленного прежнего текста (графики) не допускаются.

После внесения исправлений документ должен удовлетворять требованиям микрофильмирования, установленным ГОСТ 13.1.002.

(Измененная редакция, Изм. № 1).

6.1.6 Фамилии, названия учреждений, организаций, фирм, название изделий и другие имена собственные в отчете приводят на языке оригинала. Допускается транслитерировать имена собственные и приводить названия организаций в переводе на язык отчета с добавлением (при первом упоминании) оригинального названия.

6.1.7 Сокращение русских слов и словосочетаний в отчете — по ГОСТ 7.12.

Создание структуры документа

ГОСТ 7.32 - 2001

6.2 Построение отчета

6.2.1 Наименования структурных элементов отчета «СПИСОК ИСПОЛНИТЕЛЕЙ», «РЕФЕРАТ», «СОДЕРЖАНИЕ», «ОПРЕДЕЛЕНИЯ», «ОБОЗНАЧЕНИЯ И СОКРАЩЕНИЯ», «ВВЕДЕНИЕ», «ЗАКЛЮЧЕНИЕ», «СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ», «ПРИЛОЖЕНИЕ» служат заголовками структурных элементов отчета. Заголовки структурных элементов следует располагать в середине строки без точки в конце и печатать прописными буквами, не подчеркивая.

(Измененная редакция, Изм. № 1).

6.2.2 Основную часть отчета следует делить на разделы, подразделы и пункты. Пункты, при необходимости, могут делиться на подпункты. При делении текста отчета на пункты и подпункты необходимо, чтобы каждый пункт содержал законченную информацию.

6.2.3 Разделы, подразделы, пункты и подпункты следует нумеровать арабскими цифрами и записывать с абзацного отступа.

Разделы должны иметь порядковую нумерацию в пределах всего текста, за исключением приложений.

Пример — 1, 2, 3 и т. д.

Номер подраздела или пункта включает номер раздела и порядковый номер подраздела или пункта, разделенные точкой.

Пример — 1.1, 1.2, 1.3 и т. д.

Номер подпункта включает номер раздела, подраздела, пункта и порядковый номер подпункта, разделенные точкой.

Пример — 1.1.1.1, 1.1.1.2, 1.1.1.3 и т. д.

После номера раздела, подраздела, пункта и подпункта в тексте точку не ставят.

Если текст отчета подразделяют только на пункты, их следует нумеровать, за исключением приложений, порядковыми номерами в пределах всего отчета.

Если раздел или подраздел имеет только один пункт или пункт имеет один подпункт, то нумеровать его не следует.

6.2.4 Разделы, подразделы должны иметь заголовки. Пункты, как правило, заголовков не имеют. Заголовки должны четко и кратко отражать содержание разделов, подразделов.

6.2.5 Заголовки разделов, подразделов и пунктов следует печатать с абзацного отступа с прописной буквы без точки в конце, не подчеркивая.

Если заголовок состоит из двух предложений, их разделяют точкой.

Создание структуры документа

ГОСТ 7.32 - 2001

6.4 Нумерация разделов, подразделов, пунктов, подпунктов отчета

6.4.1 Разделы отчета должны иметь порядковые номера в пределах всего отчета, обозначенные арабскими цифрами без точки и записанные с абзацного отступа. Подразделы должны иметь нумерацию в пределах каждого раздела. Номер подраздела состоит из номеров раздела и подраздела, разделенных точкой. В конце номера подраздела точка не ставится. Разделы, как и подразделы, могут состоять из одного или нескольких пунктов.

6.4.2 Если отчет не имеет подразделов, то нумерация пунктов в нем должна быть в пределах каждого раздела, и номер пункта должен состоять из номеров раздела и пункта, разделенных точкой. В конце номера пункта точка не ставится.

Пример

1 Типы и основные размеры

1.1
1.2 } ***Нумерация пунктов первого раздела отчета***
1.3

2 Технические требования

2.1
2.2 } ***Нумерация пунктов второго раздела отчета***
2.3

Создание структуры документа

ГОСТ 7.32 - 2001

Если отчет имеет подразделы, то нумерация пунктов должна быть в пределах подраздела и номер пункта должен состоять из номеров раздела, подраздела и пункта, разделенных точками, например:

3 Методы испытаний

3.1 Аппараты, материалы и реактивы

3.1.1
3.1.2
3.1.3
} ***Нумерация пунктов первого подраздела третьего раздела отчета***

3.2 Подготовка к испытанию

3.2.1
3.2.2
3.2.3
} ***Нумерация пунктов второго подраздела третьего раздела отчета***

6.4.3 Если раздел состоит из одного подраздела, то подраздел не нумеруется. Если подраздел состоит из одного пункта, то пункт не нумеруется.

6.4.1—6.4.3 **(Измененная редакция, Изм. № 1).**

6.4.4 Если текст отчета подразделяется только на пункты, то они нумеруются порядковыми номерами в пределах всего отчета.

6.4.5 Пункты, при необходимости, могут быть разбиты на подпункты, которые должны иметь порядковую нумерацию в пределах каждого пункта, например 4.2.1.1, 4.2.1.2, 4.2.1.3 и т. д.

6.4.6 Внутри пунктов или подпунктов могут быть приведены перечисления.

Перед каждым элементом перечисления следует ставить дефис. При необходимости ссылки в тексте отчета на один из элементов перечисления вместо дефиса ставятся строчные буквы в порядке русского алфавита, начиная с буквы а (за исключением букв ё, з, й, о, ч, ъ, ы, ь).

Для дальнейшей детализации перечислений необходимо использовать арабские цифры, после которых ставится скобка, а запись производится с абзацного отступа, как показано в примере.

Пример

а) _____
б) _____
1) _____
2) _____
в) _____

(Измененная редакция, Изм. № 1).

6.4.7 Если отчет состоит из двух и более частей, каждая часть должна иметь свой порядковый номер. Номер каждой части следует проставлять арабскими цифрами на титульном листе под указанием вида отчета, например, «Часть 2».

6.4.8 Каждый структурный элемент отчета следует начинать с нового листа (страницы).

6.4.9 Нумерация страниц отчета и приложений, входящих в состав отчета, должна быть сквозная.

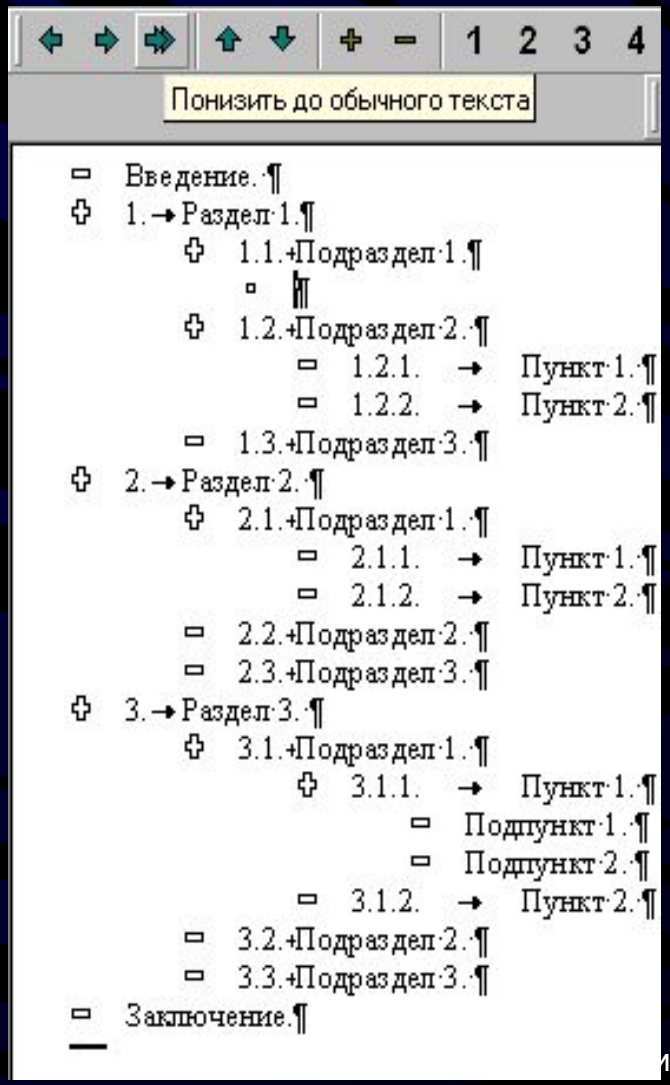
Создание структуры документа

- Вы можете сначала набрать ваш текстовый документ, а, потом организовать его структуру, но лучше сначала продумать структуру документа и следовать ей в работе

Создание структуры документа

- Введение, Заключение, Приложение, Список рисунков, Список таблиц, Список использованных источников оформляются без применения списка
- Разделы документа имеют 1-ый уровень заголовка
- Подразделы имеют 2-ой уровень заголовка и аналогичный вид многоуровневого списка
- Пункты имеют 3–й уровень заголовка и аналогичный вид многоуровневого списка

Создание структуры документа



Создание структуры документа

6.14 Приложения

6.14.1 Приложение оформляют как продолжение данного документа на последующих его листах или выпускают в виде самостоятельного документа.

6.14.2 В тексте отчета на все приложения должны быть даны ссылки. Приложения располагают в порядке ссылок на них в тексте отчета.

6.14.3 Каждое приложение следует начинать с новой страницы с указанием наверху посередине страницы слова «Приложение», его обозначения.

Приложение должно иметь заголовок, который записывают симметрично относительно текста с прописной буквы отдельной строкой.

6.14.4 Приложения обозначают заглавными буквами русского алфавита, начиная с А, за исключением букв Ё, З, Й, О, Ч, Ъ, Ы, Ь. После слова «Приложение» следует буква, обозначающая его последовательность.

Допускается обозначение приложений буквами латинского алфавита, за исключением букв I и O.

В случае полного использования букв русского и латинского алфавитов допускается обозначать приложения арабскими цифрами.

Если в отчете одно приложение, оно обозначается «Приложение А».

6.14.2—6.14.4 **(Измененная редакция, Изм. № 1).**

6.14.5 Текст каждого приложения, при необходимости, может быть разделен на разделы, подразделы, пункты, подпункты, которые нумеруют в пределах каждого приложения. Перед номером ставится обозначение этого приложения.

6.14.6 Приложения должны иметь общую с остальной частью документа сквозную нумерацию страниц.

6.14.7 Приложение или несколько приложений могут быть оформлены в виде отдельной книги отчета, при этом на титульном листе под номером книги следует писать слово «Приложение». При необходимости такое приложение может иметь раздел «Содержание».

(Измененная редакция, Изм. № 1).

ГОСТ 7.32 - 2001

6.2. Установка параметров страницы

6.1.2 Отчет о НИР должен быть выполнен любым печатным способом на пишущей машинке или с использованием компьютера и принтера на одной стороне листа белой бумаги формата А4 через полтора интервала. Цвет шрифта должен быть черным, высота букв, цифр и других знаков — не менее 1,8 мм (кегль не менее 12). Полужирный шрифт не применяется.

Текст отчета следует печатать, соблюдая следующие размеры полей: правое — не менее 10 мм, верхнее и нижнее — не менее 20 мм, левое — не менее 30 мм.

Разрешается использовать компьютерные возможности акцентирования внимания на определенных терминах, формулах, теоремах, применяя шрифты разной гарнитуры.

ГОСТ 7.32 - 2001

Установка параметров страницы

- Перед тем как набирать текст нового документа необходимо проверить устраивают ли вас отступы слева, справа, сверху и снизу от краев листа до набираемого вами текста
- Существуют определенные стандарты отступов при оформлении различных документов

Установка параметров страницы

- Для установления отступов используется меню Файл команда Параметры страницы вкладка Поля
- Вкладка Размер бумаги позволяет выбрать формат листа и определить его ориентацию – книжную или альбомную

6.3. Разработка и создание колонтитулов документа

6.3 Нумерация страниц отчета

6.3.1 Страницы отчета следует нумеровать арабскими цифрами, соблюдая сквозную нумерацию по всему тексту отчета. Номер страницы проставляют в центре нижней части листа без точки.

6.3.2 Титульный лист включают в общую нумерацию страниц отчета. Номер страницы на титульном листе не проставляют.

6.3.3 Иллюстрации и таблицы, расположенные на отдельных листах, включают в общую нумерацию страниц отчета.

Иллюстрации и таблицы на листе формата А3 учитывают как одну страницу.

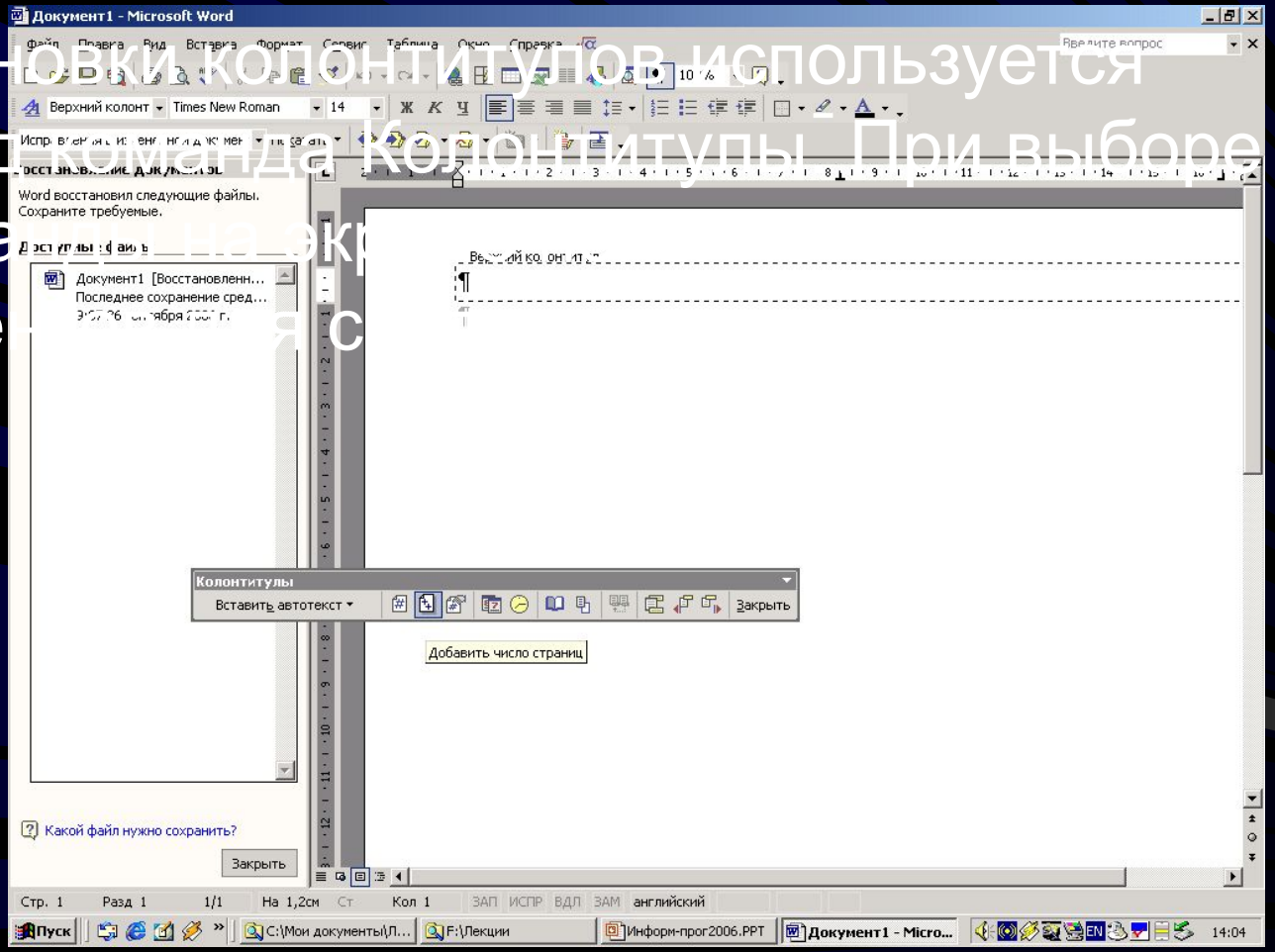
ГОСТ 7.32 - 2001

Разработка и создание колонтитулов документа

- Колонтитул — это текст и/или рисунок, который печатается внизу или вверху каждой страницы документа
- В зависимости от места расположения (на верхнем или на нижнем поле страницы) колонтитулы бывают верхними и нижними
- Колонтитулы могут быть различными для четных и нечетных страниц, а также колонтитул первого листа может отличатся от колонтитулов остальных страниц

Разработка и создание колонтитулов документа

- Для установки колонтитулов используется меню Вид команда Колонтитулы. При выборе этой кома экр инструме й с



6.4. Вставка сносок

6.7.4 При необходимости дополнительного пояснения в отчете его допускается оформлять в виде сноски. Знак сноски ставят непосредственно после того слова, числа, символа, предложения, к которому дается пояснение. Знак сноски выполняют надстрочно арабскими цифрами со скобкой. Допускается вместо цифр выполнять сноски звездочками «*». Применять более трех звездочек на странице не допускается.

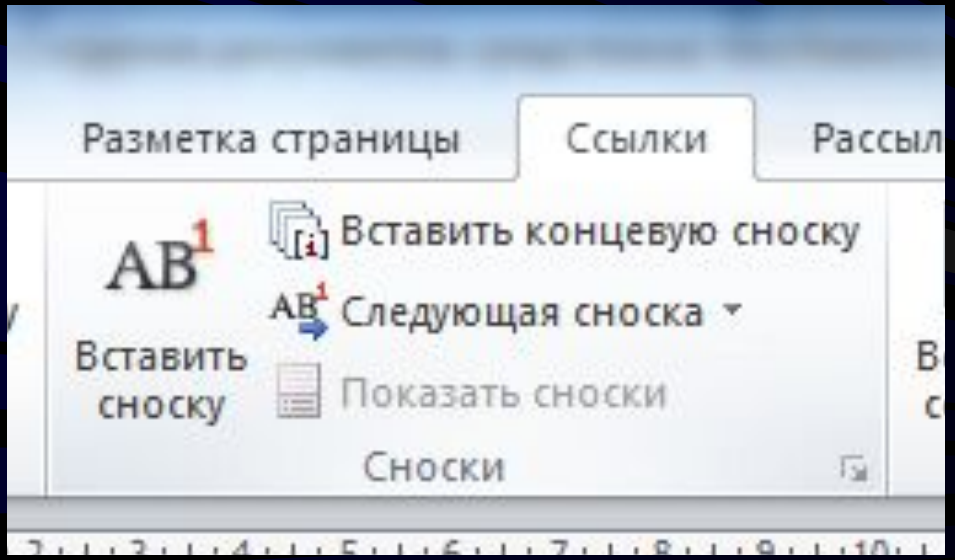
Сноску располагают в конце страницы с абзацного отступа, отделяя от текста короткой горизонтальной линией слева. Сноску к таблице располагают в конце таблицы над линией, обозначающей окончание таблицы.

(Введен дополнительно, Изм. № 1).

ГОСТ 7.32 - 2001

Вставка сносок

- Установите курсор туда, куда следует вставить знак сноски
- Выберите команду Сноска в меню Вставка
- Выберите Обычную или Концевую



Вставка сносок

- Введите текст сноски в область сносок. Для возврата к основному тексту документа щелкните его
- Чтобы перенести, скопировать или удалить сноску, надо работать со знаком сноски, а не с текстом в области сносок
- При перемещении, копировании и удалении знака сноски автоматически выполняется перенумерация всех сносок

6.5 Создание формульных выражений

6.8 Формулы и уравнения

6.8.1 Уравнения и формулы следует выделять из текста в отдельную строку. Выше и ниже каждой формулы или уравнения должно быть оставлено не менее одной свободной строки. Если уравнение не умещается в одну строку, то оно должно быть перенесено после знака равенства (=) или после знаков плюс (+), минус (—), умножения (х), деления (:) или других математических знаков, причем знак в начале следующей строки повторяют. При переносе формулы на знаке, символизирующем операцию умножения, применяют знак «Х».

6.8.2 Пояснение значений символов и числовых коэффициентов следует приводить непосредственно под формулой в той же последовательности, в которой они даны в формуле.

6.8.3 Формулы в отчете следует нумеровать порядковой нумерацией в пределах всего отчета арабскими цифрами в круглых скобках в крайнем правом положении на строке.

Пример

$$A = a:b. \qquad (1)$$

Одну формулу обозначают — **(1)**.

6.8.4 Формулы, помещаемые в приложениях, должны нумероваться отдельной нумерацией арабскими цифрами в пределах каждого приложения с добавлением перед каждой цифрой обозначения приложения, например формула (В.1).

6.8.5 Ссылки в тексте на порядковые номера формул дают в скобках. Пример — ... в формуле (1).

6.8.6 Допускается нумерация формул в пределах раздела. В этом случае номер формулы состоит из номера раздела и порядкового номера формулы, разделенных точкой, например (3.1).

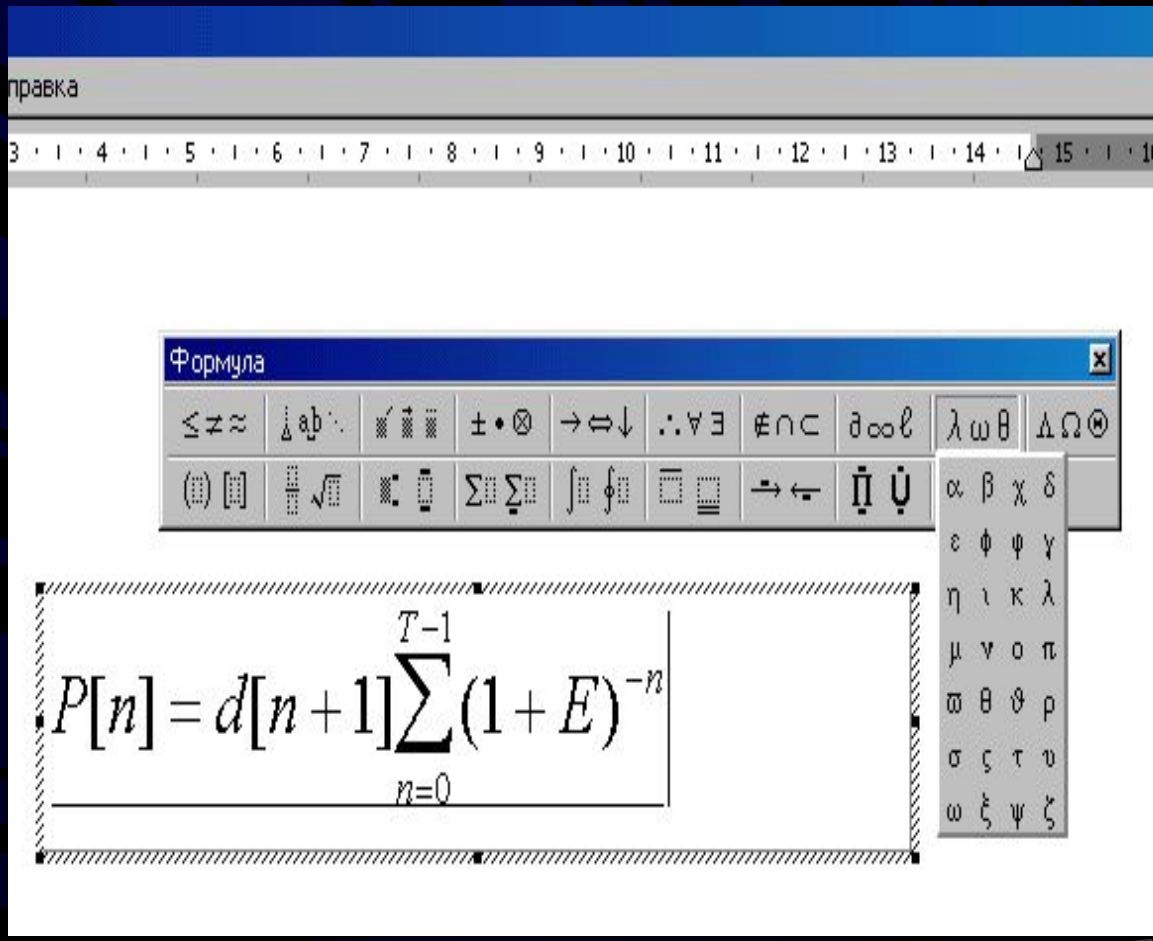
6.8.7 Порядок изложения в отчете математических уравнений такой же, как и формул.

6.8.8 В отчете допускается выполнение формул и уравнений рукописным способом черными чернилами.

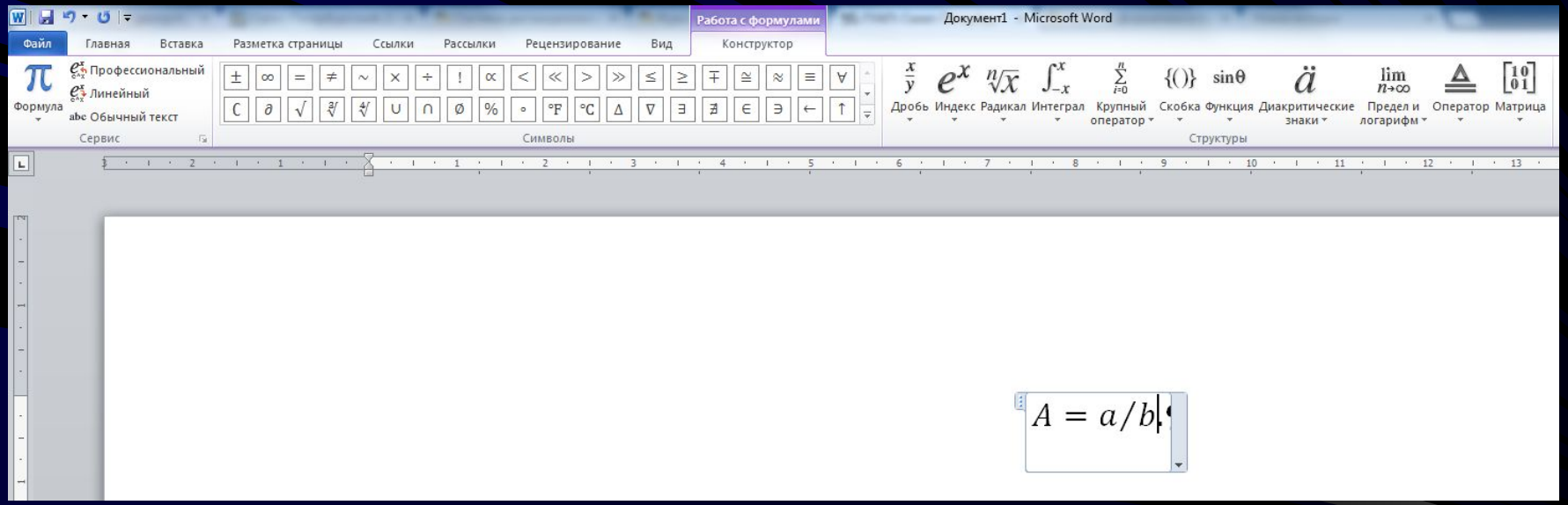
ГОСТ 7.32 - 2001

Создание формульных выражений

- Удобно для создания формул применять специальную программу Microsoft Equation.



Создание формульных выражений



6.6. Разработка и создание таблиц

ГОСТ 7.32 - 2001

6.6 Таблицы

6.6.1 Таблицы применяют для лучшей наглядности и удобства сравнения показателей. Наименование таблицы, при его наличии, должно отражать ее содержание, быть точным, кратким. Наименование таблицы следует помещать над таблицей слева, без абзацного отступа в одну строку с ее номером через тире.

(Измененная редакция, Изм. № 1).

6.6.2 Таблицу следует располагать в отчете непосредственно после текста, в котором она упоминается впервые, или на следующей странице.

6.6.3 На все таблицы должны быть ссылки в отчете. При ссылке следует писать слово «таблица» с указанием ее номера.

6.6.4 Таблицу с большим числом строк допускается переносить на другой лист (страницу). При переносе части таблицы на другой лист (страницу) слово «Таблица», ее номер и наименование указывают один раз слева над первой частью таблицы, а над другими частями также слева пишут слова «Продолжение таблицы» и указывают номер таблицы.

Таблицу с большим количеством граф допускается делить на части и помещать одну часть под другой в пределах одной страницы. Если строки и графы таблицы выходят за формат страницы, то в первом случае каждой части таблицы повторяется головка, во втором случае — боковик. При делении таблицы на части допускается ее головку или боковик заменять соответственно номером граф и строк. При этом нумеруют арабскими цифрами графы и (или) строки первой части таблицы.

Если повторяющийся в разных строках графы таблицы текст состоит из одного слова, то его после первого написания допускается заменять кавычками; если из двух и более слов, то при первом повторении его заменяют словами «То же», а далее — кавычками. Ставить кавычки вместо повторяющихся цифр, марок, знаков, математических и химических символов не допускается. Если цифровые или иные данные в какой-либо строке таблицы не приводят, то в ней ставят прочерк.

6.6.5 Цифровой материал, как правило, оформляют в виде таблиц. Пример оформления таблицы приведен на рисунке 1.

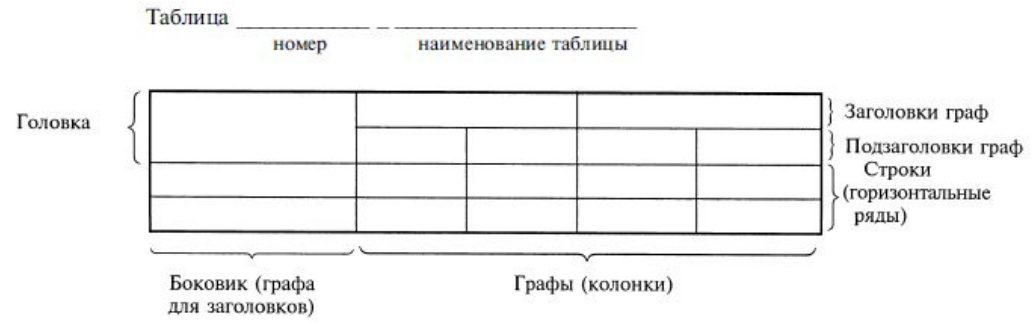


Рисунок 1

Разработка и создание таблиц

6.6.6 Таблицы, за исключением таблиц приложений, следует нумеровать арабскими цифрами сквозной нумерацией.

Допускается нумеровать таблицы в пределах раздела. В этом случае номер таблицы состоит из номера раздела и порядкового номера таблицы, разделенных точкой.

Таблицы каждого приложения обозначают отдельной нумерацией арабскими цифрами с добавлением перед цифрой обозначения приложения.

Если в отчете одна таблица, то она должна быть обозначена «Таблица 1» или «Таблица В.1», если она приведена в приложении В.

6.6.4—6.6.6 **(Измененная редакция, Изм. № 1).**

6.6.7 Заголовки граф и строк таблицы следует писать с прописной буквы в единственном числе, а подзаголовки граф — со строчной буквы, если они составляют одно предложение с заголовком, или с прописной буквы, если они имеют самостоятельное значение. В конце заголовков и подзаголовков таблиц точки не ставят.

6.6.8 Таблицы слева, справа и снизу, как правило, ограничивают линиями. Допускается применять размер шрифта в таблице меньший, чем в тексте.

Разделять заголовки и подзаголовки боковика и граф диагональными линиями не допускается.

Горизонтальные и вертикальные линии, разграничивающие строки таблицы, допускается не проводить, если их отсутствие не затрудняет пользование таблицей.

Заголовки граф, как правило, записывают параллельно строкам таблицы. При необходимости допускается перпендикулярное расположение заголовков граф.

Головка таблицы должна быть отделена линией от остальной части таблицы.

Разработка и создание таблиц

- Вы можете создавать таблицу любым способом, но при этом она должна быть отформатирована, иметь заголовок (и нумерацию, если таких таблиц несколько)

Разработка и создание таблиц

- Каждая таблица должна иметь название. Пример названия:

Таблица 8 – Пример расчета заработной платы

- В тексте документа должна быть *ссылка* на таблицу, например:

Результаты расчета заработной платы сотрудников представлены в таблице 8.

- Первая ссылка на таблицу (ссылок может быть несколько) обязательно должна быть *до* первого появления таблицы в тексте.

6.7. Разработка и создание иллюстраций

ГОСТ 7.32 - 2001

6.5 Иллюстрации

6.5.1 Иллюстрации (чертежи, графики, схемы, компьютерные распечатки, диаграммы, фотоснимки) следует располагать в отчете непосредственно после текста, в котором они упоминаются впервые, или на следующей странице.

Иллюстрации могут быть в компьютерном исполнении, в том числе и цветные.

На все иллюстрации должны быть даны ссылки в отчете.

6.5.2 Чертежи, графики, диаграммы, схемы, иллюстрации, помещаемые в отчете, должны соответствовать требованиям государственных стандартов Единой системы конструкторской документации (ЕСКД).

Допускается выполнение чертежей, графиков, диаграмм, схем посредством использования компьютерной печати.

6.5.3 Фотоснимки размером меньше формата А4 должны быть наклеены на стандартные листы белой бумаги.

6.5.4 Иллюстрации, за исключением иллюстрации приложений, следует нумеровать арабскими цифрами сквозной нумерацией.

Если рисунок один, то он обозначается «Рисунок 1». Слово «рисунок» и его наименование располагают посередине строки.

6.5.5 Допускается нумеровать иллюстрации в пределах раздела. В этом случае номер иллюстрации состоит из номера раздела и порядкового номера иллюстрации, разделенных точкой. Например, Рисунок 1.1.

6.5.6 Иллюстрации, при необходимости, могут иметь наименование и пояснительные данные (подрисуночный текст). Слово «Рисунок» и наименование помещают после пояснительных данных и располагают следующим образом: Рисунок 1 — Детали прибора.

6.5.7 Иллюстрации каждого приложения обозначают отдельной нумерацией арабскими цифрами с добавлением перед цифрой обозначения приложения. Например, Рисунок А.3.

6.5.8 При ссылках на иллюстрации следует писать «...в соответствии с рисунком 2» при сквозной нумерации и «...в соответствии с рисунком 1.2» при нумерации в пределах раздела.

Разработка и создание иллюстраций

- Под рисунком понимается все, что не является формулой или таблицей
- Обычно в качестве рисунков выступают графики, диаграммы, фотографии, структуры, алгоритмы и т.п.

Разработка и создание иллюстраций

- Вы можете создавать рисунок любым способом, в том числе и средствами Word, однако использование Word для создания рисунков *нежелательно*, поскольку оно ведет к некоторым проблемам в тексте при автоматическом форматировании документа
- Рисунок может быть создан любыми другими программными средствами, а позднее скопирован в нужное место документа

Разработка и создание иллюстраций

- Каждый рисунок должен иметь *под*рисуночную подпись. Пример *под*рисуночной подписи:

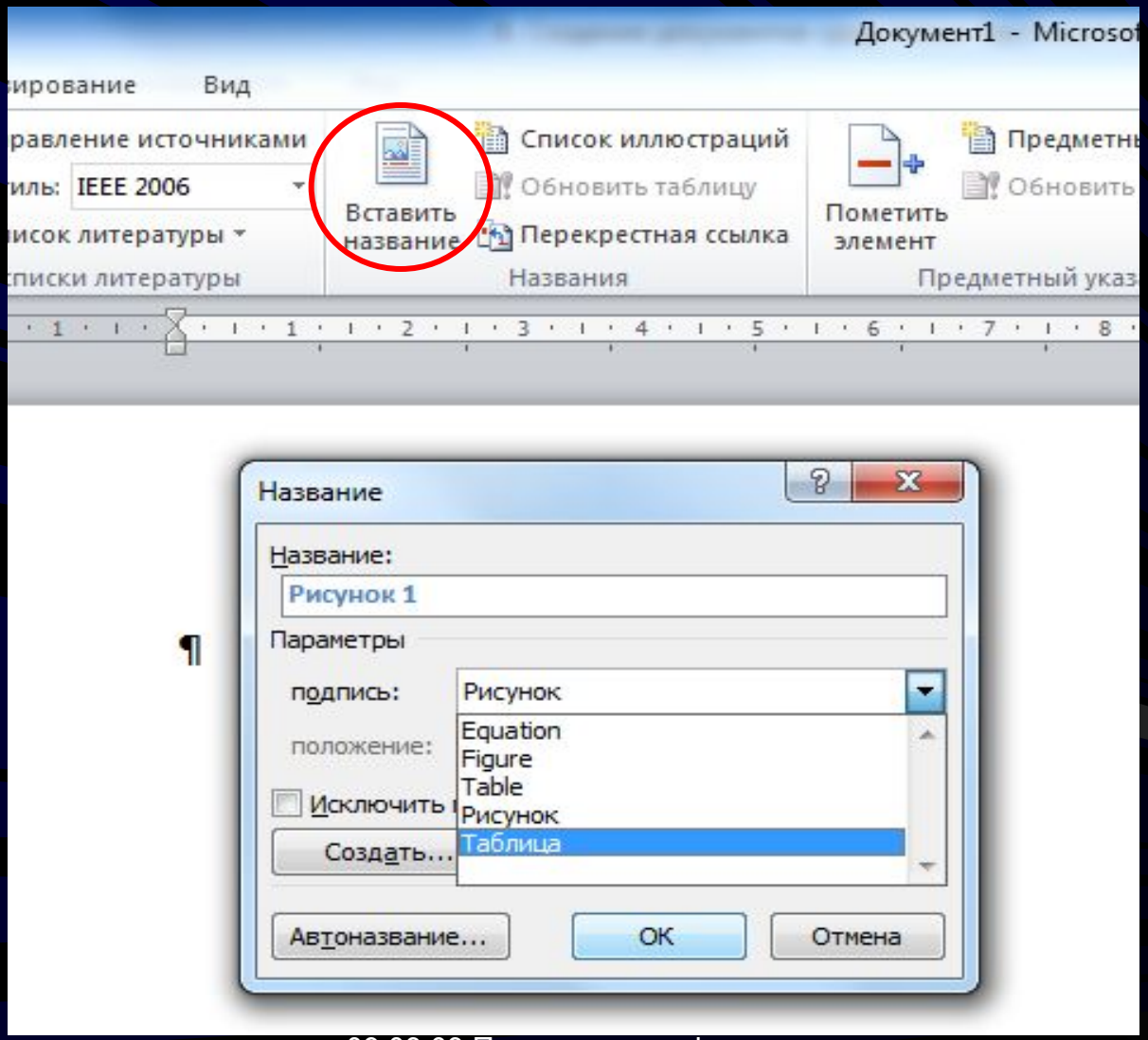
Рисунок 4 – Структура современной информатики как науки

- В тексте документа должна быть *ссылка* на рисунок, например:

Как показано на рисунке 4, современная информатика как наука состоит из двух составляющих: теоретической и прикладной

- Первая ссылка на рисунок (ссылок может быть несколько) обязательно должна быть *до* первого появления рисунка в тексте.

Разработка и создание иллюстраций



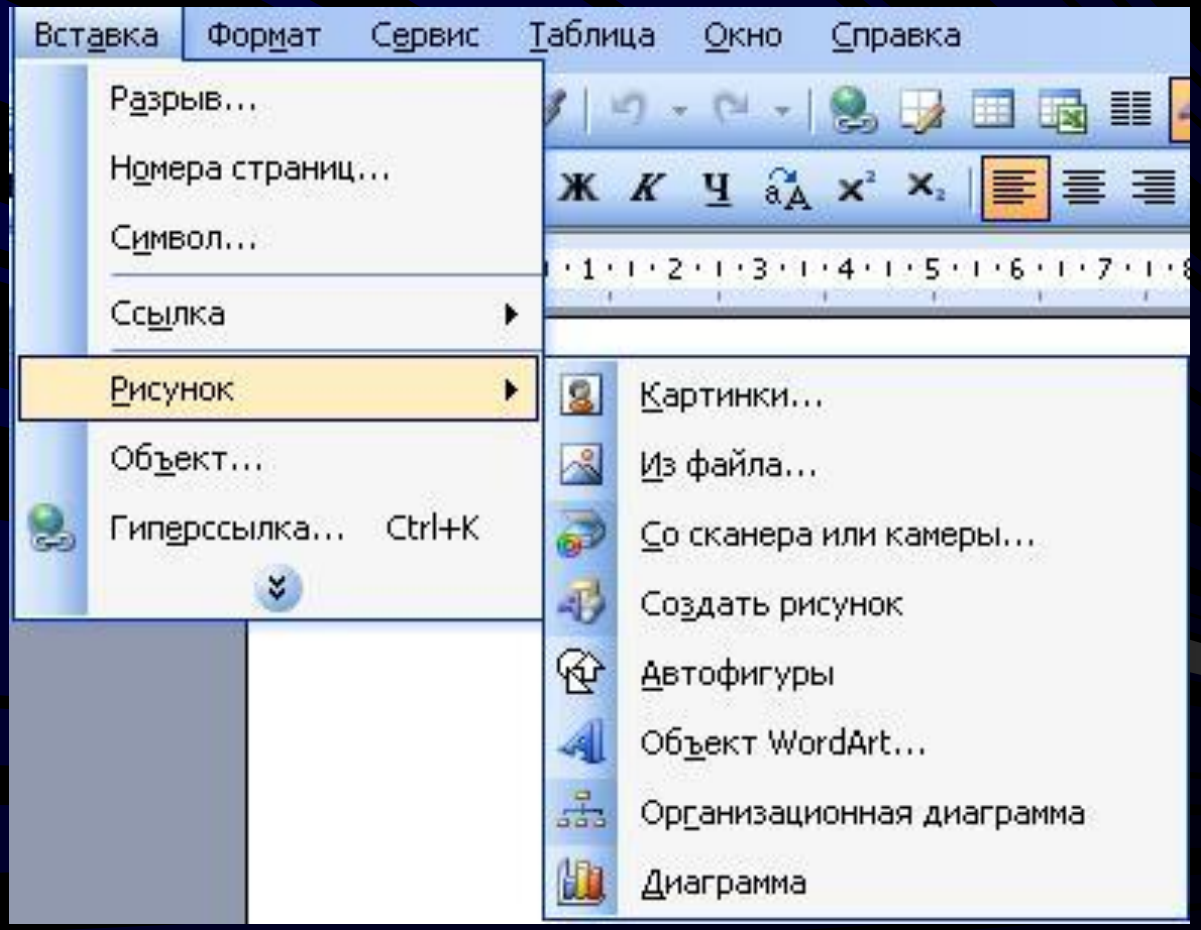
Оформление документов рисунками

- Вставка готовых рисунков
  - Из файла
  - Из коллекции картинок
- Создать рисунок средствами текстового процессора
- Отредактировать готовый рисунок средствами текстового процессора

Для создания рисунков удобнее использовать специализированные средства

Вставка готовых рисунков (1)

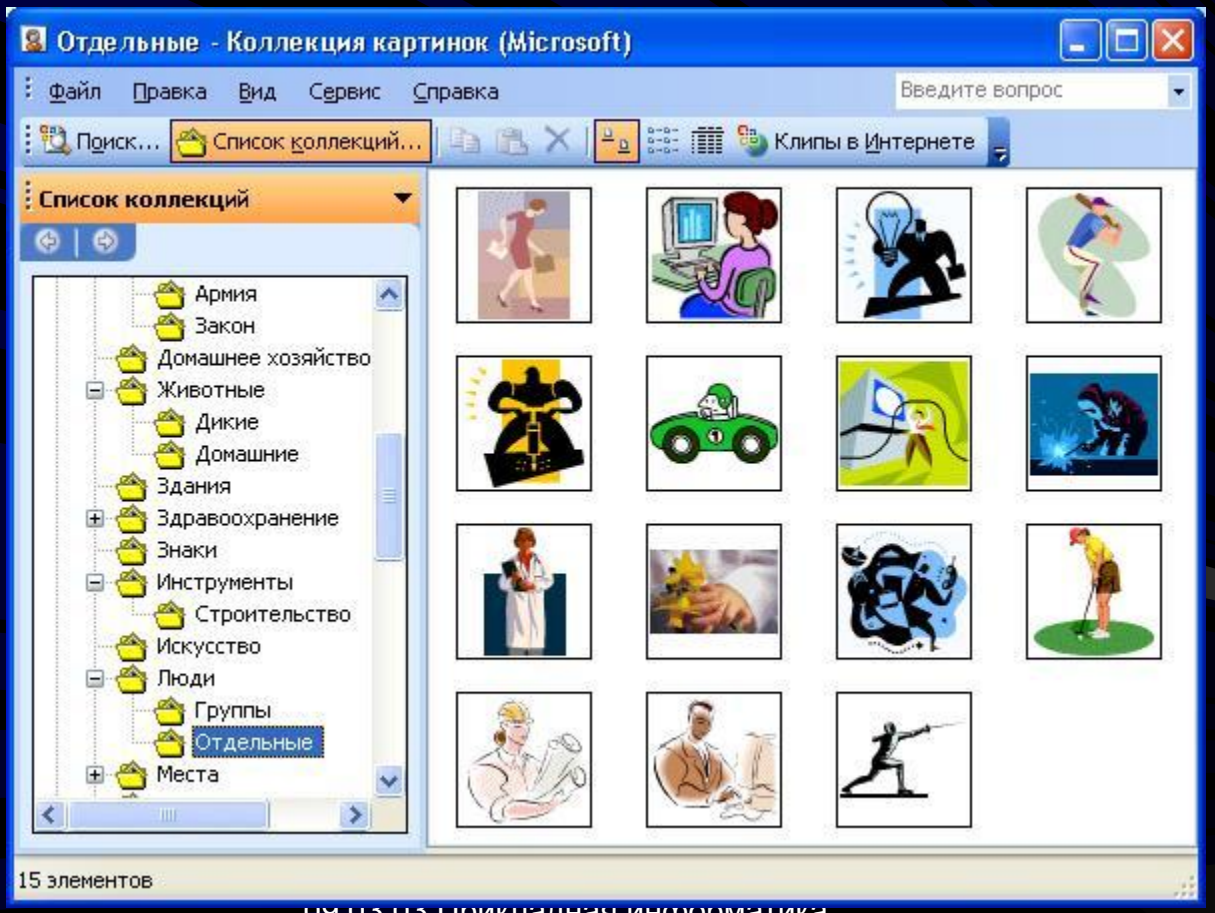
- с помощью меню **Вставка – Рисунок Картинки**
  Из файла …



Вставка готовых рисунков (2)

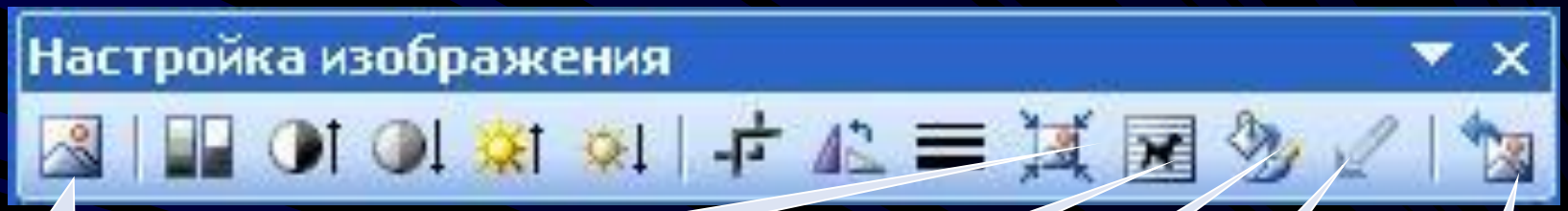
с помощью меню **Вставка – Рисунок Картинки**

Из коллекции MS Office



Редактирование готовых рисунков

С помощью панели



Добавить рисунок

Сжатие рисунков

Обтекание текстом

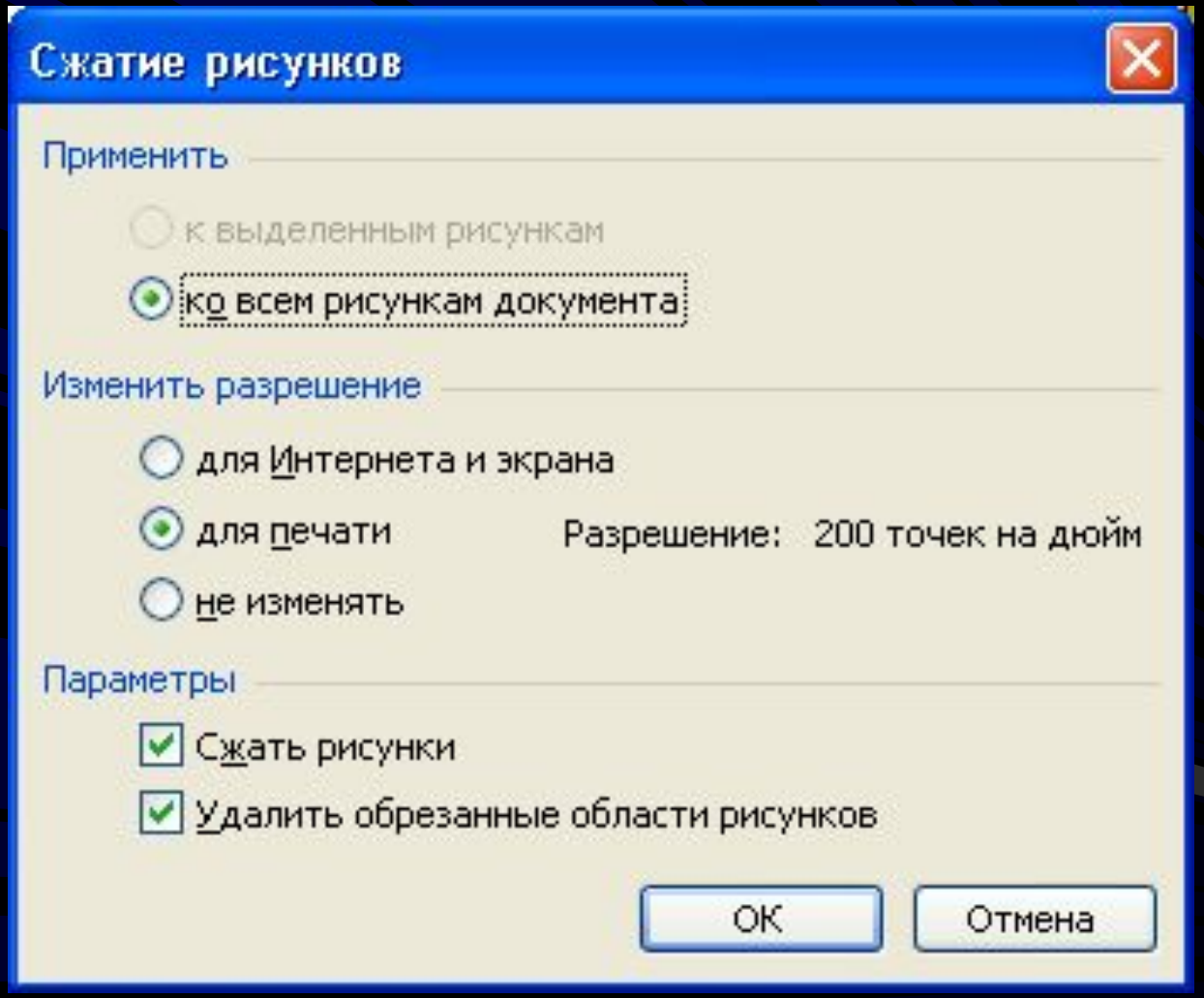
Формат объекта

Прозрачный цвет

Сброс параметров

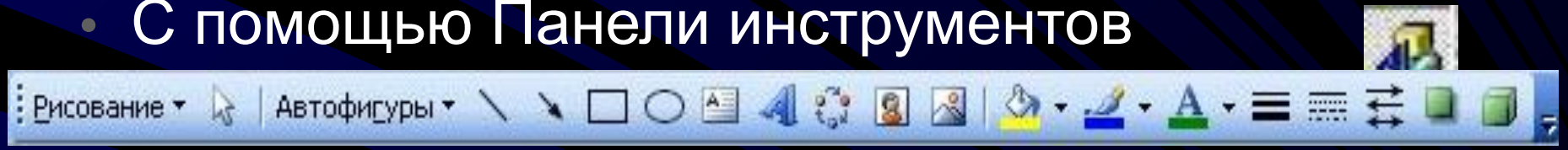
Обрезка

Сжатие рисунков



Создание рисунков средствами текстового процессора

- С помощью Панели инструментов

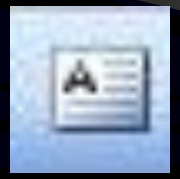


- **Вид –Панели инструментов Рисование**

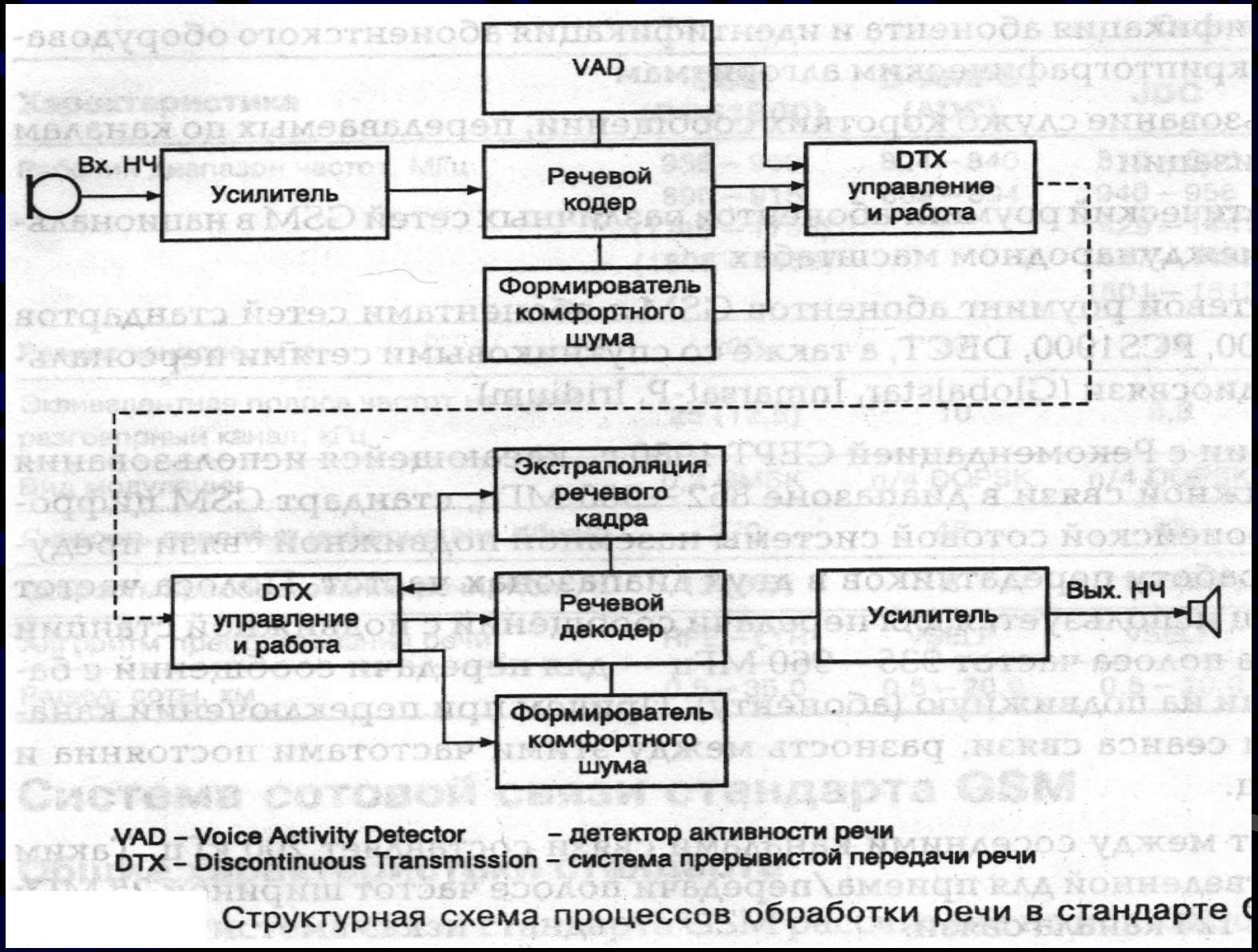
Рекомендации: Активно используйте

- Группировку
- Копирование
- Поворот
- Отражение
- Для вставки сложных условных обозначений

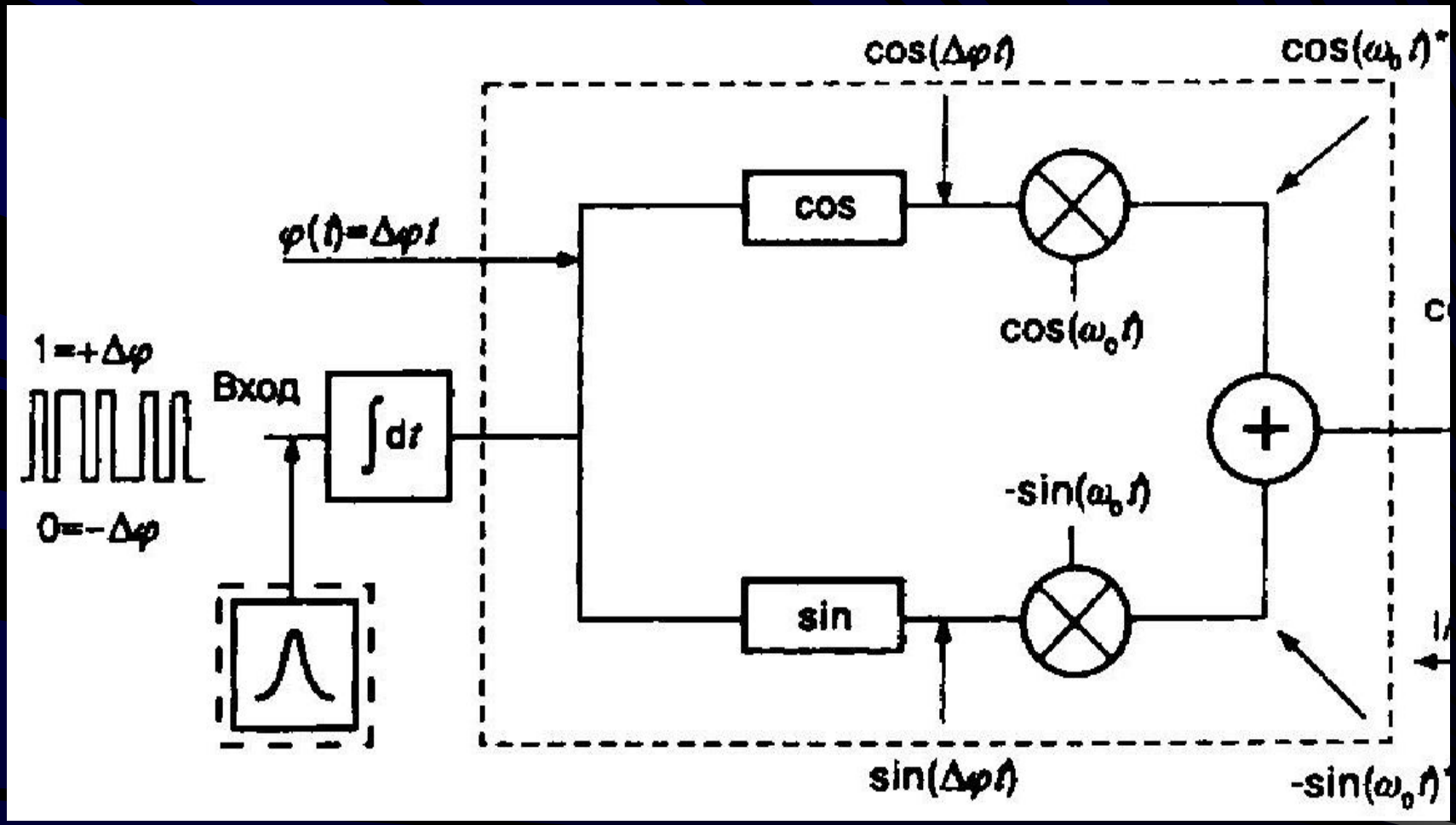
– редактор формул в элементе «Надпись»



Для создания рисунков удобнее использовать специализированные средства



Структурная схема процессов обработки речи в стандарте


cos(Δφ·t)
cos(ω0·t)
cos
φ(t)=Δφ·t
cos(ω0·t)
1=+Δφ
Вход
∫dt
0=-Δφ
-sin(ω0·t)
sin
sin(Δφ·t)
-sin(ω0·t)

Оформление рисунков в документе

- Завершающий этап создания рисунка:
  - Группировка
  - Настройка взаимодействия с текстом (Формат – Рисунок (Объект) – вкладка Положение)
- Все иллюстрации называются рисунками и могут нумероваться в пределах каждого раздела:

Рисунок 2.1

- Рисунки располагаются по тексту после первой ссылки на него. Если есть дальнейшая ссылка, то

«см. рисунок 2.1».

- При необходимости рисунок может иметь наименование и поясняющие сведения, которые помещаются под рисунком после указания его номера:

Рисунок 2.1 - Функциональная схема блока селектора

Вставка объекта WordArt

Информационные технологии

Информационные технологии

Информационные технологии

Информационные технологии

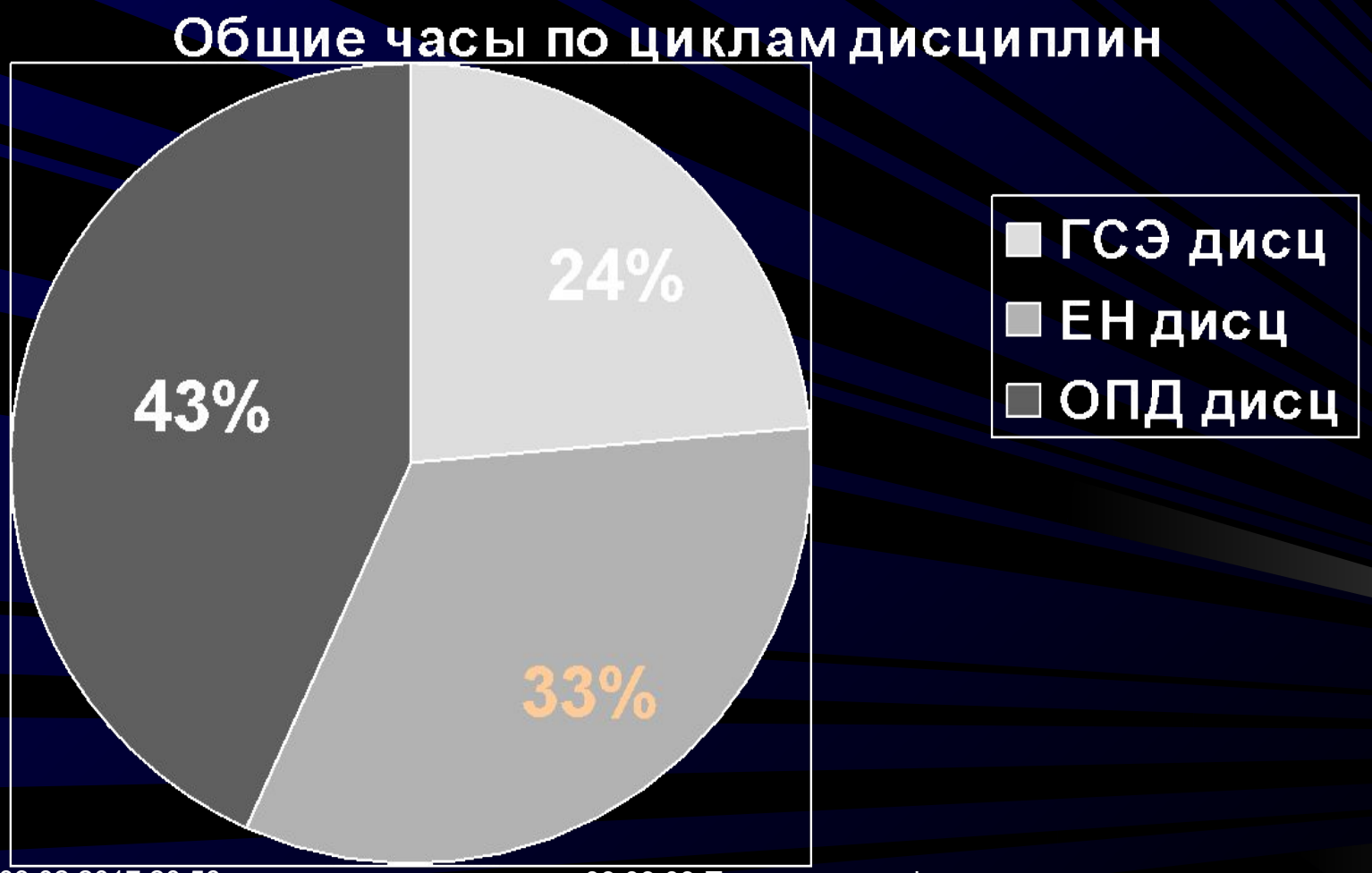
Вставка диаграмм

Виды диаграмм

- **Гистограмма**
- **Круговая**
- **Линейчатая**
- **Точечная**
- **Гладкий график**
- **Организационная**
- **И др.**

Гистограмма

Круговая диаграмма



Разработка и создание рисунков

- Вы можете создавать рисунок любым способом, в том числе и средствами Word, однако использование Word для создания рисунков нежелательно, поскольку оно ведет к некоторым проблемам в тексте при автоматическом форматировании документа
- Рисунок может быть создан любыми другими программными средствами, а позднее скопирован в нужное место документа

6.8. Вставка ссылок на литературу

6.9 Ссылки

6.9.1 Ссылки на использованные источники следует указывать порядковым номером библиографического описания источника в списке использованных источников. Порядковый номер ссылки заключают в квадратные скобки. Нумерация ссылок ведется арабскими цифрами в порядке приведения ссылок в тексте отчета независимо от деления отчета на разделы.

(Измененная редакция, Изм. № 1).

6.9.2 **(Исключен, Изм. № 1).**

6.9.3 При ссылках на стандарты и технические условия указывают только их обозначение, при этом допускается не указывать год их утверждения при условии полного описания стандарта и технических условий в списке использованных источников в соответствии с ГОСТ 7.1.

(Измененная редакции, Изм. № 1).

6.9.4 **(Исключен, Изм. № 1).**

6.13 Список использованных источников

Сведения об источниках следует располагать в порядке появления ссылок на источники в тексте отчета и нумеровать арабскими цифрами без точки и печатать с абзацного отступа.

ГОСТ 7.32 - 2001

Вставка ссылок на литературу

- Ссылки на литературу являются принципиально необходимой составляющей любого документа и предназначены для указания заимствования информации из других источников (книг, статей, сайтов и т.п.)
- Позволить себе не пользоваться ссылками на литературу можно только в том случае, когда вы претендуете на *абсолютную новизну* текста

Вставка ссылок на литературу

- В технической литературе принято вставлять ссылки в текст документа в виде числа в квадратных скобках, например [35]
- Число в квадратных скобках есть номер цитируемого источника в *списке использованных источников*, который помещается в конце документа
- В зависимости от вида документа этот *список* может иметь название Указатель литературы, Литература, Список литературы, *Список использованных источников* и т.п. В каждом конкретном случае это название необходимо уточнить

Вставка ссылок на литературу

- *Список использованных источников* должен составляться в порядке цитирования,
- Если *список* составляется в порядке цитирования, то первая *ссылка* в тексте делается на первый литературный источник [1], вторая на второй [2] и т.п. Допускаются многократные *ссылки* на один и тот же источник, однако вторая *ссылка* первый раз все равно должна появиться после первой

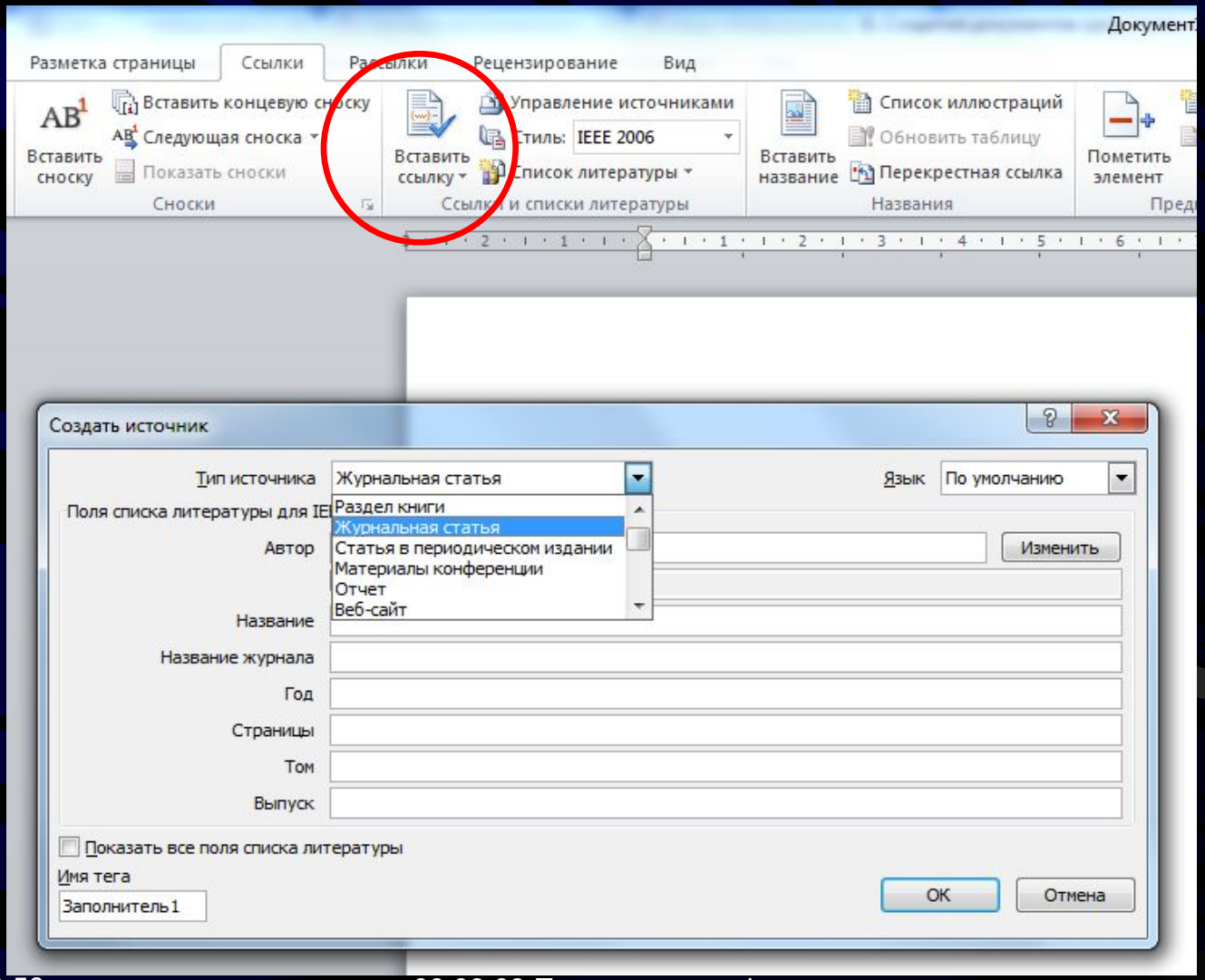
Вставка ссылок на литературу

- В соответствии с требованиями ВАК при оформлении диссертаций в случае большого объема списка использованных источников допускается составление его по алфавиту. Сами ссылки на источники в тексте оформляются также в виде номера позиции источника в квадратных скобках

Вставка ссылок на литературу

- Word позволяет автоматически нумеровать и сортировать список литературы
- Изменения в списке литературы приводят и к изменениям ссылок
- Если список литературы выполнен в виде нумерованного списка, а ссылки реализованы через режим перекрестных ссылок Word, то можно автоматически изменить номера ссылок

Вставка ссылок на литературу



6.9. Использование закладок

- Закладкой называется специальная пометка фрагмента документа, позволяющая обратиться к нему из любой другой части документа и, например, вставить этот фрагмент в нужное место
- Когда закладка создана, в любом другом месте документа можно установить Перекрестную ссылку на любую из имеющихся закладок
- Изменение текста закладки после выполнения операции обновления документа приводит к изменению фрагмента, вставленного по ссылке в текст

Использование закладок

- Чтобы создать закладку, надо выделить фрагмент текста и выбрать пункт меню Вставка строка Закладки. Открывшееся окно диалога попросит вас ввести имя закладки
- Целесообразно выбирать осмысленное имя, чтобы потом легко узнавать, к чему собственно относится закладка, однако при этом надо иметь в виду, что пробелы в имени закладки не допускаются
- Перекрестная ссылка имеет несколько вариантов, выбираемых из меню

6.10. Составление оглавления, списка таблиц и иллюстраций

5.4 Содержание

5.4.1 Содержание включает введение, наименование всех разделов, подразделов, пунктов (если они имеют наименование), заключение, список использованных источников и наименование приложений с указанием номеров страниц, с которых начинаются эти элементы отчета о НИР.

5.4.2 При составлении отчета, состоящего из двух и более частей, в каждой из них должно быть свое содержание. При этом в первой части помещают содержание всего отчета с указанием номеров частей, в последующих — только содержание соответствующей части. Допускается в первой части вместо содержания последующих частей указывать только их наименования.

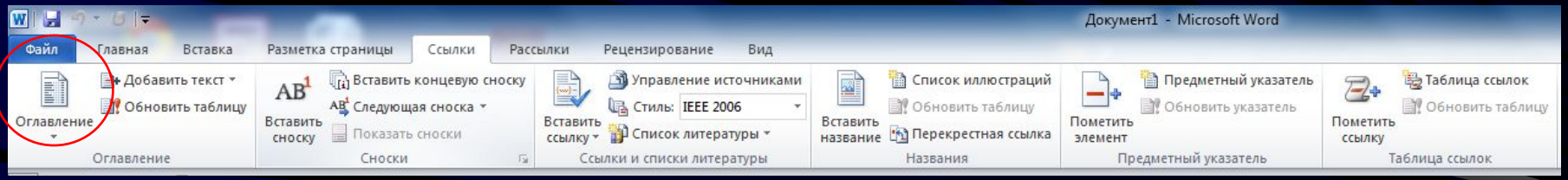
5.4.3 В отчете о НИР объемом не более 10 страниц содержание допускается не составлять.

5.5—5.5.3 (Исключены, Изм. № 1).

ГОСТ 7.32 - 2001

Составление оглавления, списка таблиц и иллюстраций

- После этого в пункте Вставка главного меню надо выбрать режим Оглавление и указатели. В открывшейся таблице надо выбрать соответствующую вкладку, нажать кнопку Параметры и задать уровень стилей заголовков документа, которые используются при построении оглавления, после чего нажать кнопку ОК



6.11. Титульные листы и бланки задания

5.1 Титульный лист

5.1.1 Титульный лист является первой страницей отчета о НИР и служит источником информации, необходимой для обработки и поиска документа.

5.1.2 На титульном листе приводят следующие сведения:

- наименование вышестоящей организации;
- наименование организации-исполнителя НИР;
- индекс Универсальной десятичной классификации (УДК);
- коды Высших классификационных группировок Общероссийского классификатора промышленной и сельскохозяйственной продукции для НИР (ВКГОКП), предшествующих постановке продукции на производство;
- номера, идентифицирующие отчет;
- грифы согласования и утверждения;
- наименование работы;
- наименование отчета;
- вид отчета (заключительный, промежуточный);

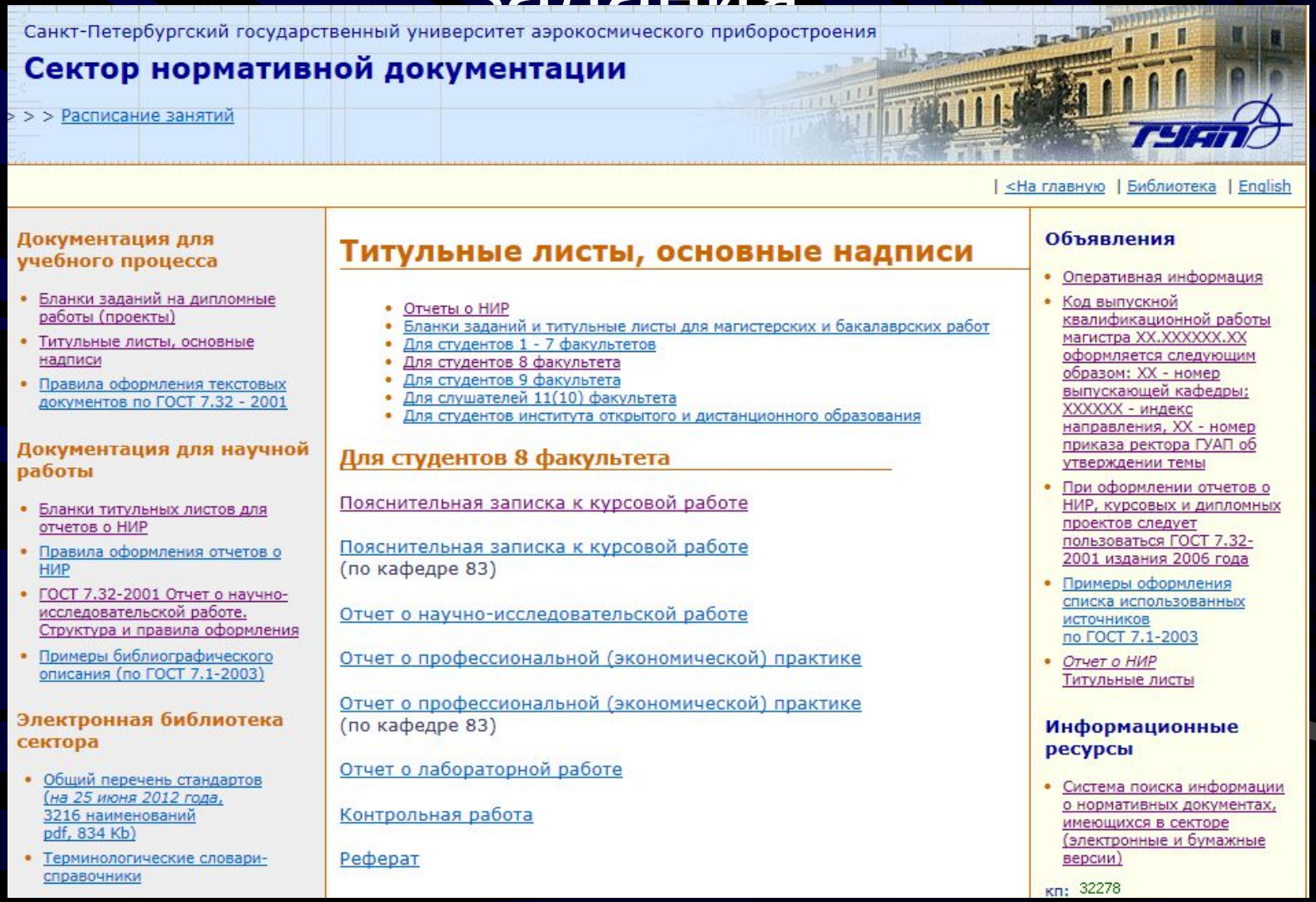
- номер (шифр) работы;
- должности, ученые степени, ученые звания, фамилии и инициалы руководителей организации-исполнителя НИР, руководителей НИР;
- место и дату составления отчета.

5.1.3 Если отчет о НИР состоит из двух и более частей, то каждая часть должна иметь свой титульный лист, соответствующий титульному листу первой части и содержащий сведения, относящиеся к данной части.

5.1.4 Титульный лист следует оформлять в соответствии с 6.10.

ГОСТ 7.32 - 2001

Титульные листы и бланки задания



Образцы титульных листов учебных работ http://guap.ru/guap/standart/titl_main.shtml

До новых встреч